Accepted in IEEE Transactions on Autonomous Mental Development, 2013.

Published:
Murata S., Namikawa J., Arie H., Sugano S., Tani J.: "Learning to reproduce fluctuating time series by inferring their time-dependent stochastic properties: application in robot learning via tutoring", IEEE Trans. on Autonomous Mental Development, Vol. 5, No 4, pp. 298-310, 2013

Learning to Reproduce Fluctuating Time Series by Inferring Their Time-dependent Stochastic Properties: Application in Robot Learning via Tutoring

Shingo Murata, Jun Namikawa, Hiroaki Arie, Shigeki Sugano, and Jun Tani *

Abstract

This study proposes a novel type of dynamic neural network model that can learn to extract stochastic or fluctuating structures hidden in time series data. The network learns to predict not only the mean of the next input state, but also its time-dependent variance. The training method is based on maximum likelihood estimation by using the gradient descent method, and the likelihood function is expressed as a function of the estimated variance. Regarding the model evaluation, we present numerical experiments in which training data were generated in different ways of utilizing Gaussian noise. Our analysis showed that the network can predict the time-dependent variance and the mean, as well as that it can reproduce the target stochastic sequence data by utilizing the estimated variance. Furthermore, it was shown that a humanoid robot using the proposed network can learn to reproduce latent stochastic structures hidden in fluctuating tutoring trajectories. This learning scheme is essential for the acquisition of sensory-guided skilled behavior.

1 Introduction

The ability to learn to predict perceptual outcomes of intended actions has been considered to be essential for the developmental learning of actions in both infants [1] and artificial agents [2,3]. Among various connectionist models, recurrent neural networks (RNNs) have been intensively investigated for their suitability for prediction by learning [4–6]. In the context of behavior learning for robots, Tani and colleagues have shown that RNN-based models can learn to predict perceptual consequences of actions in navigation problems [7], as well as to predict perceptual sequences for sets of action intentions in object manipulation tasks [8–10]. RNN-based models, however, are considerably limited due to the deterministic nature of their prediction mechanism. As deterministic dynamical systems, RNNs cannot learn to extract stochastic structures hidden in noisy time series data used for training. If RNNs are forced to learn such time series data, the learning process tends to become unstable with the accumulation of errors.

To address this problem, Namikawa and colleagues recently proposed a novel continuous-time RNN (CTRNN) model that can learn to predict not only the next mean state, but also the variance of the observable variables at each time step [11]. The predicted variance functions

*This study was supported by grants from the Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore (AH/OCL/1082/0111/I2R).
Jun Tani is the corresponding author.
S. Murata, H. Arie, and S. Sugano are with the Department of Modern Mechanical Engineering, Waseda University, Tokyo, Japan (e-mail: {murata, arie}@sugano.mech.waseda.ac.jp; sugano@waseda.jp).
J. Namikawa is with the Brain Science Institute, RIKEN, Saitama, Japan (e-mail: jnamika@gmail.com).
J. Tani is with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea (e-mail: tani1216jp@gmail.com).

as an inverse weighting factor for the prediction error that is back-propagated in the process of learning. The formulation of the model is analogous to the free energy minimization principle proposed by Friston [12, 13], in which learning, generation, and recognition of stochastic sequences are formulated by means of likelihood maximization.

This article presents the complete formulation of this novel model referred to as stochastic CTRNN (S-CTRNN) together with the results of numerical experiments and a subsequent analysis of the dynamic characteristics of the network developed via the learning process. Furthermore, we describe how the proposed model can be successfully applied in robot learning problems dealing with fluctuating training data by conducting an experiment on sensory-guided robot behavior demonstrated to a robot by human trainers.

Different approaches for estimating and utilizing variance for robot behavior learning have been proposed, including combinations of Gaussian mixture model (GMM) and Gaussian mixture regression (GMR) [14–16]. We consider that the implementation of a method for the estimation or prediction of variance in the CTRNN model can be used as an alternative to the previously proposed approaches.

The next section presents details about the forward dynamics, training, and generation method of S-CTRNN.

2 Neural Network Model

2.1 Overview

A number of researchers have investigated the learning characteristics of CTRNN models [17–19] for deterministic continuous-time data trajectories. The S-CTRNN proposed in the present article makes use of a novel feature manifested by "variance prediction units" allocated in the output layer. By utilizing these units, the network predicts not only the mean of the next input, but also its variance. In this method, the mean and the variance can be obtained by means of maximizing the likelihood function for the sequence data. Furthermore, upon achieving convergence of the likelihood, the network can reproduce sequences with the same stochastic properties as the training data by adding Gaussian noise with the variance predicted at each time step to the predicted mean and subsequently feeding these values as input. The details of the model scheme are described in the following section.

2.2 Forward Dynamics

Figure 1 presents a block diagram of the S-CTRNN model which can predict the variance for each output neuron.

The internal state of the ith neuron at time step t ($u_{t,i}$) is updated in accordance with

$$u_{t,i} = \begin{cases} \left(1 - \dfrac{1}{\tau_i}\right) u_{t-1,i} + \dfrac{1}{\tau_i} \left(\displaystyle\sum_{j \in I_I} w_{ij} x_{t,j} + \sum_{j \in I_C} w_{ij} c_{t-1,j} + b_i \right) & (1 \leq t \wedge i \in I_C), \\ \displaystyle\sum_{j \in I_C} w_{ij} c_{t,j} + b_i & (1 \leq t \wedge i \in I_O \cup I_V), \end{cases} \tag{1}$$

where I_C, I_O, and I_V are the neuron index sets, τ_i is the time constant of the ith neuron, $w_{i,j}$ is the weight of the connection between the jth and the ith neuron, $c_{t,j}$ is the activation value of the jth context neuron at time step t, $x_{t,j}$ is the jth external input at time step t, and b_i is the bias of the ith neuron.

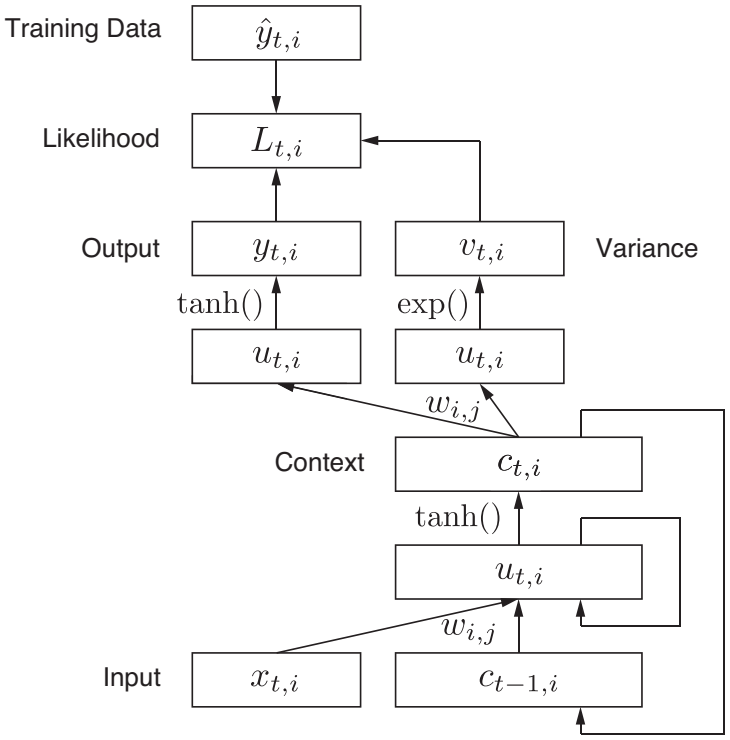

Figure 1: Block diagram of the S-CTRNN model. S-CTRNN consists of input, context, output, and variance units. Using the recorded training sequences, the network is trained offline. The predicted variances are then used for calculation of the likelihood function.

The respective activation values of context unit $c_{t,i}$, output unit $y_{t,i}$, and variance unit $v_{t,i}$ are calculated as follows:

$$c_{t,i} = \tanh(u_{t,i}) \qquad (0 \leq t \wedge i \in I_C), \tag{2}$$

$$y_{t,i} = \tanh(u_{t,i}) \qquad (1 \leq t \wedge i \in I_O), \tag{3}$$

$$v_{t,i} = \exp(u_{t,i}) \qquad (1 \leq t \wedge i \in I_V). \tag{4}$$

2.3 Training Method

The network is trained through maximum likelihood estimation by utilizing the gradient descent method [20]. The training process consists of the following phases.

1. Initialization of learnable parameters (weight, bias, and initial states) (described in 2.4).
2. Generation of output and variance by forward dynamics under the current parameter settings (described in 2.2).
3. Calculation of likelihood by using training data, generated output and variance predicted by the network.
4. Updating the parameters with the gradient descent method.
5. Repeating 2) - 4) until prediction error converges (described in 2.5).

The current section introduces phases 3) and 4).

Here, the learnable parameters of the network are denoted by $\boldsymbol{\theta}$. Let $X = (\boldsymbol{x}_t)_{t=1}^{T}$ be an input sequence, where T is the length of the sequence. In this case, the probability density function of training data $\hat{y}_{t,i}$ is defined as

$$p(\hat{y}_{t,i} \mid X, \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi v_{t,i}}} \exp\left(-\frac{(y_{t,i} - \hat{y}_{t,i})^2}{2v_{t,i}}\right), \tag{5}$$

where $y_{t,i}$ and $v_{t,i}$ are the outputs generated by the network and $\hat{y}_{t,i}$ is the training data. This equation is derived with the assumption that the observable data sequence is embedded into additive Gaussian noise. It is well known that minimizing the mean square error is equivalent to maximizing the likelihood determined from a normal distribution. Therefore, (5) is a natural extension of the training of the neural network.

The likelihood function L_{out} parameterized by $\boldsymbol{\theta}$, is denoted by

$$L_{\text{out}} = \prod_{t=1}^{T} \prod_{i \in I_O} p(\hat{y}_{t,i} \mid X, \boldsymbol{\theta}). \tag{6}$$

The network generates an estimate of the prediction error in the form of a variance $v_{t,i}$. The network can avoid unstable learning sequences since the variance functions as an inverse weighting factor for the square error $(y_{t,i} - \hat{y}_{t,i})^2$. More specifically, the effect of the prediction error is reduced when the variance is large (as the error is divided by the variance), whereas the effect is increased when the variance is small. Therefore, the extent of error back-propagation can be autonomously reduced in the case of learning parts of time sequences which display considerable fluctuation. This relaxes the predictive learning of stochastic sequences.

The training method involves choosing the most appropriate value for the parameter $\boldsymbol{\theta}$ by maximizing the likelihood L_{out}. More precisely, we use the gradient descent method with a momentum term as the training procedure. Here, the logarithm of the expression in (6) is used to facilitate the calculation.

$$\ln L_{\text{out}} = \sum_{t=1}^{T} \sum_{i \in I_O} \left(-\frac{\ln(2\pi v_{t,i})}{2} - \frac{(y_{t,i} - \hat{y}_{t,i})^2}{2v_{t,i}}\right). \tag{7}$$

The model parameters at step n ($\boldsymbol{\theta}(n)$) of the training process are updated in accordance with

$$\boldsymbol{\theta}(n) = \boldsymbol{\theta}(n-1) + \Delta\boldsymbol{\theta}(n), \tag{8}$$

$$\Delta\boldsymbol{\theta}(n) = \alpha\left(\frac{\partial \ln L_{\text{out}}}{\partial \boldsymbol{\theta}} + \eta \Delta\boldsymbol{\theta}(n-1)\right), \tag{9}$$

where α is the learning rate and η is a parameter representing the momentum term. The partial differential equations $\frac{\partial \ln L_{\text{out}}}{\partial \boldsymbol{\theta}}$ for each learnable parameter, which can be solved by a conventional back-propagation through time method [21], are given by

$$\frac{\partial \ln L_{\text{out}}}{\partial w_{ij}} = \begin{cases} \dfrac{1}{\tau_i} \displaystyle\sum_{t=1}^{T} x_{t,j} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}} & (i \in I_C \wedge j \in I_I), \\ \dfrac{1}{\tau_i} \displaystyle\sum_{t=1}^{T} c_{t-1,j} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}} & (i \in I_C \wedge j \in I_C), \\ \displaystyle\sum_{t=1}^{T} c_{t,j} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}} & (i \in I_O \cup I_V \wedge j \in I_C), \end{cases} \tag{10}$$

$$\frac{\partial \ln L_{\text{out}}}{\partial b_i} = \begin{cases} \dfrac{1}{\tau_i} \displaystyle\sum_{t=1}^{T} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}} & (i \in I_C), \\ \displaystyle\sum_{t=1}^{T} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}} & (i \in I_O \cup I_V), \end{cases} \tag{11}$$

$$\frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}} = \begin{cases} \left(1 - c_{t,i}^2\right) \left\{ \displaystyle\sum_{k \in I_C} \frac{w_{ki}}{\tau_k} \frac{\partial \ln L_{\text{out}}}{\partial u_{t+1,k}} + \sum_{k \in I_O \cup I_V} w_{ki} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,k}} \right\} \\ + \left(1 - \dfrac{1}{\tau_i}\right) \dfrac{\partial \ln L_{\text{out}}}{\partial u_{t+1,i}} & (0 \le t \wedge i \in I_C), \\ -\dfrac{y_{t,i} - \hat{y}_{t,i}}{v_{t,i}} \left(1 - y_{t,i}^2\right) & (1 \le t \wedge i \in I_O), \\ -\dfrac{1}{2} + \dfrac{(y_{t,i} - \hat{y}_{t,i})^2}{2v_{t,i}} & (1 \le t \wedge i \in I_V). \end{cases} \tag{12}$$

In the current study, the likelihood (or the error) was back-propagated to the initial time step without a truncation depth.

Although we have presented only the case in which the training data set is a single sequence, the method can be easily extended to training involving several sequences by using the sum of the gradients for each sequence.

When several sequences are used as training data, an initial state must be provided for each sequence. We consider that the distribution of the initial states conforms to a normal distribution. The probability density function for $u_{0,i}^{(s)}$, which is the initial state of the ith neuron corresponding to the sth training sequence, is defined as

$$p(u_{0,i}^{(s)} \mid \sigma, \hat{u}_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\hat{u}_i - u_{0,i}^{(s)})^2}{2\sigma^2} \right), \tag{13}$$

where σ^2 is the predefined variance and $\hat{u}_i$ is the mean value of the initial states, which is a learnable parameter.

The likelihood function L_{init} parameterized by $\hat{u}_i$ and $u_{0,i}^{(s)}$ is given by

$$L_{\text{init}} = \prod_{s \in I_S} \prod_{i \in I_C} p(u_{0,i}^{(s)} \mid \sigma, \hat{u}_i), \tag{14}$$

where I_S is the sequence index set. The logarithm of the expression in (14) is used to facilitate the calculation.

$$\ln L_{\text{init}} = \sum_{s \in I_S} \sum_{i \in I_C} \left(-\frac{\ln(2\pi\sigma^2)}{2} - \frac{(u_{0,i}^{(s)} - \hat{u}_i)^2}{2\sigma^2} \right). \tag{15}$$

The initial states are updated to maximize the likelihood $\ln L_{\text{all}}$, which is the sum of $\ln L_{\text{out}}$ and $\ln L_{\text{init}}$. The partial differential equations for each parameter are given by

$$\frac{\partial \ln L_{\text{all}}}{\partial \hat{u}_i} = \sum_{s \in I_S} \frac{1}{\sigma^2} (u_{0,i}^{(s)} - \hat{u}_i), \tag{16}$$

$$\frac{\partial \ln L_{\text{all}}}{\partial u_{0,i}^{(s)}} = \frac{\partial \ln L_{\text{out}}}{\partial u_{0,i}^{(s)}} - \frac{1}{\sigma^2} (u_{0,i}^{(s)} - \hat{u}_i). \tag{17}$$

2.4 Parameter Setting for Training

All biases and connection weights were initialized with randomly chosen values from a uniform distribution on the interval $\left[-\frac{1}{M}, \frac{1}{M}\right]$, where M is the number of context neurons. Each initial internal state $u_{0,i}^{(s)}$ and the mean value of the initial states $\hat{u}_i$ were initialized with a value of 0. Since the maximum value of L_{out} depends on the total length T_{total} of the training sequences and the dimensionality d of the output neurons, the learning rate α was scaled by a parameter $\tilde{\alpha}$ satisfying the relation $\alpha = \frac{1}{T_{\text{total}} d}\tilde{\alpha}$. In all experiments presented here, $\tilde{\alpha}$ and the momentum term were chosen to be $\tilde{\alpha} = 0.0001$ and $\eta = 0.9$, respectively. The remaining parameters, including the number of context neurons, the time constant, and the variance of the initial states, are introduced in the respective section of each experiment.

2.5 Error Evaluation

In order to evaluate the error of the trained network, we computed the following mean square error (MSE) for each sequence s,

$$E^{(s)} = \frac{1}{2Td} \sum_{t=1}^{T} \sum_{i=1}^{d} (y_{t,i}^{(s)} - \hat{y}_{t,i}^{(s)})^2, \tag{18}$$

where $y_{t,i}^{(s)}$ is the output and $\hat{y}_{t,i}^{(s)}$ is the training data corresponding to the sth sequence. Although there is not an explicit MSE value as a criterion for determining the success or failure of training, it converges a certain value that depends on the stochastic properties of the training data, especially values for the noise variance. For example, the MSE corresponding to the training data with larger noise variance converges to a larger value than that of the training data with smaller noise variance. The MSE after training is described in the respective section of each experiment.

2.6 Generation Method

After achieving convergence for the likelihood (or the error) through the training process, the network was able to predict the input state for the next time step from the current input state. There are the following two different ways to feed the current input $x_{t,i}$ into the network:

$$x_{t,i} = \begin{cases} \hat{y}_{t-1,i}, & \text{(19a)} \\ y_{t-1,i} + \epsilon_{t-1,i}, & \text{(19b)} \end{cases}$$

where $\hat{y}_{t-1,i}$ is an external input state representing a recorded training state or actual sensory information acquired by the robot, $y_{t-1,i}$ is an output state or a predicted input state generated by the trained network, and $\epsilon_{t-1,i}$ is Gaussian noise given by

$$\epsilon_{t-1,i} = \epsilon(v_{t-1,i}), \tag{20}$$

where $\epsilon(\sigma^2)$ is a Gaussian noise generator with a zero mean and a standard deviation of σ (Fig. 2). In (19a), the current input is the current external input, and this case is referred to as the "open-loop mode." On the other hand, in (19b), the current input is derived from the predicted mean value to which the noise estimated at the previous step is added, and this case is referred to as the "closed-loop mode with the addition of estimated noise." Equation (19b) can autonomously generate stochastic sequences whose ensembles are assumed to reproduce the stochastic structure of the training data.

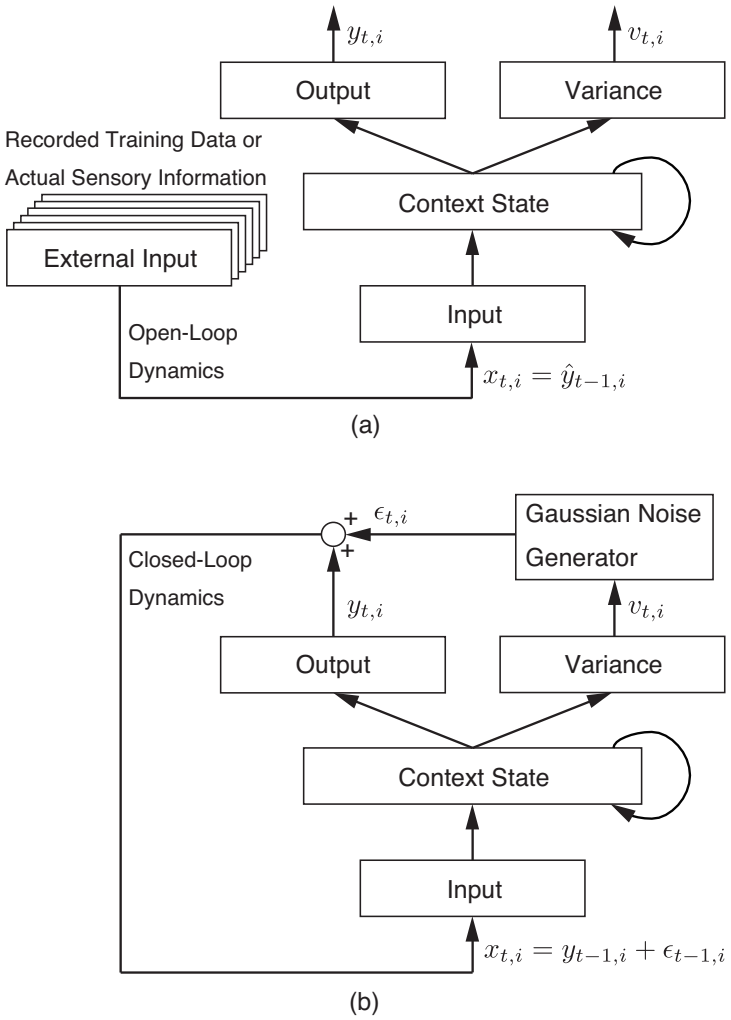

Figure 2: Generation method: (a) open-loop mode and (b) closed-loop mode with the addition of estimated noise.

3 Numerical Experiments

To demonstrate how S-CTRNN learns to extract stochastic dynamic structures hidden in training data, we conducted three numerical experiments with various values for the variance of the added Gaussian noise. The first experiment, described in Section 3.1, involved learning a one-dimensional stochastic sine curve with added Gaussian noise with a time-dependent variance in order to test whether the network was capable of predicting the noise variance in a simple setting. The second experiment, described in Section 3.2, involved learning stochastic Lissajous curves with multiple constant values for the noise variance. In this experiment, we tested whether a set of stochastic spatiotemporal patterns could be reproduced as multiple attractors of the trained network.

In these experiments, Gaussian noise was added to the reference trajectories of the sinusoidal or Lissajous patterns at each time step. In Section 3.3, we present the results for a case in which the target patterns were generated by attractor dynamics, namely a one-dimensional limit cycle, which is perturbed by time-dependent noise.

3.1 Experiment 1: A Stochastic Sine Curve with Time-Dependent Noise Variance

The training sequence was a one-dimensional sine curve with a period $P = 25$. We added Gaussian noise $\hat{\epsilon}_t = \epsilon(\hat{\sigma}_t^2)$ to the sine curve at each time step as follows:

$$\hat{y}_t = 0.8 \sin\left(\frac{2\pi t}{P}\right) + \hat{\epsilon}_t. \tag{21}$$

Here, $\hat{\sigma}_t$ is the standard deviation of the added Gaussian noise, which is defined by

$$\hat{\sigma}_t = \begin{cases} \dfrac{0.12}{P}\left(t - nP\right) + 0.01 \\ \qquad\qquad \left(nP \le t < \dfrac{(2n+1)}{2}P\right), \\ -\dfrac{0.12}{P}\left(t - (n+1)P\right) + 0.01 \\ \qquad\qquad \left(\dfrac{(2n+1)}{2}P \le t < (n+1)P\right), \end{cases} \tag{22}$$

where n is a non-negative integer (i.e., $n = \{0, 1, 2, \cdots\}$). In this task, we used one training sequence with a length $T = 1000$ $(= 40P)$.

The number of context neurons, the time constant, and the variance of the initial states were chosen to be $M = 10$, $\tau = 2$, and $\sigma^2 = 1$, respectively. We trained the network for 100,000 training steps, and the MSE converged to 0.000951. Time series were then generated with the closed-loop dynamics with addition of the estimated noise by following Eq. (19b).

The result of generating time series is shown in Fig. 3. As can be seen from Fig. 3, the output of the trained network reproduces the stochastic structure of the training sequence, which consists of a periodic pattern with a time-dependent variance. Furthermore, the predicted variance of the trained network v_t is close to the true value $\hat{v}_t = \hat{\sigma}_t^2$.

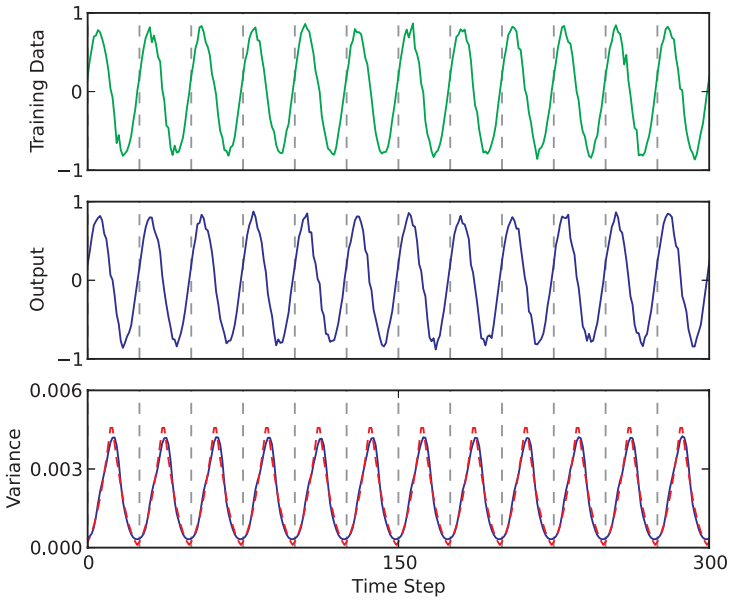

Figure 3: Generation of time series. Training sequence (first row), output of the trained network with closed-loop dynamics with addition of the estimated noise (second row), and variance predicted by the trained network (third row). In the third row (time series of variance), the blue line indicates the variance v_t predicted by the trained network, and the red dashed line indicates its true value $\hat{v}_t$.

3.2 Experiment 2: Stochastic Lissajous Curves with Multiple Constant Values for Noise Variance

The training sequences in this case were 12 Lissajous curves with a period $P = 25$, which were generated as combinations of the following sine curves $Y_{t,i}$.

$$\begin{cases} Y_{t,1} = -0.4\left(\cos\left(\dfrac{2\pi t}{P}\right) - 1\right), \\ Y_{t,2} = 0.8\sin\left(\dfrac{2\pi t}{P}\right), \\ Y_{t,3} = -0.4\left(\cos\left(\dfrac{4\pi t}{P}\right) - 1\right), \\ Y_{t,4} = 0.8\sin\left(\dfrac{4\pi t}{P}\right). \end{cases} \tag{23}$$

We added Gaussian noise $\hat{\epsilon}_{t,i}^{(s)} = \epsilon(\{\hat{\sigma}_i^{(s)}\}^2)$ to each Lissajous curve at each time step. The value of the standard deviation of the added Gaussian noise for both axes of each pattern was the same, and it was defined as

$$\hat{\sigma}_1^{(s)} = \hat{\sigma}_2^{(s)} = \begin{cases} 0.01 & (s \in 4n+1), \\ 0.03 & (s \in 4n+2), \\ 0.05 & (s \in 4n+3), \\ 0.07 & (s \in 4n+4), \end{cases} \tag{24}$$

where n is non-negative integer smaller than 3, that is, $n = \{0, 1, 2\}$.

For example, training sequences were expressed as follows (refer to the Appendix for details

about all expressions):

$$\begin{aligned}(\hat{y}_{t,1}^{(1)}, \hat{y}_{t,2}^{(1)}) &= (Y_{t,1} + \hat{\epsilon}_{t,1}^{(1)}, Y_{t,2} + \hat{\epsilon}_{t,2}^{(1)}),\\ &\vdots\\ (\hat{y}_{t,1}^{(12)}, \hat{y}_{t,2}^{(12)}) &= (Y_{t,2} + \hat{\epsilon}_{t,1}^{(12)}, -Y_{t,3} + \hat{\epsilon}_{t,2}^{(12)}).\end{aligned} \tag{25}$$

In the task, we used a training data set of 12 Lissajous curves with a length $T = 1000$ ($= 40P$). It was considered that those 12 Lissajous patterns can be embedded as multiple attractors in a single S-CTRNN by self-determining the corresponding initial context states.

The number of context neurons, the time constant, and the variance of the initial states were chosen to be $M = 60$, $\tau = 2$, and $\sigma^2 = 100$, respectively. Here, in order to embed multiple attractors into a single network, a large value was chosen for the variance of the initial states. We trained the network for 500,000 training steps, and each averaged MSE of the 4 groups consisting of 3 sequences whose standard deviation is the same value (refer to (24)), converged to 0.0000482 ($s \in 4n+1$), 0.000432 ($s \in 4n+2$), 0.00126 ($s \in 4n+3$), and 0.00233 ($s \in 4n+1$). Time series were then generated with the closed-loop dynamics with addition of the estimated noise by using (19b) with the corresponding initial states $u_{0,i}^{(s)}$.

Figure 4 presents phase plots of the training data, the output of the trained network, and two selected context neurons. By comparing both sets of the phase plots of the training data and the output in Fig. 4, we can see that the trained network produces stochastic sequences similar to the training data. Because all 12 patterns were reproduced stably by starting from the corresponding initial context states, it is considered that they were successfully embedded as multiple attractors. On the other hand, it appears that the context states lacked such stochastic properties corresponding to the training data. This finding is revisited in the discussion section.

The time series of the predicted variances and their true values for each pattern are shown in Fig. 5. Although the values of the variance $v_t^{(s)}$ predicted by the trained network oscillate, we can see that they are close to the true value $\hat{v}_t^{(s)} = \{\hat{\sigma}_t^{(s)}\}^2$. In order to evaluate the accuracy of the predicted variance, we computed the following temporal mean of predicted variance $\overline{v}_i^{(s)}$ of the trained network for each sequence s,

$$\overline{v}_i^{(s)} = \frac{1}{T}\sum_{t=1}^{T} v_{t,i}^{(s)}, \tag{26}$$

where $T = 1000$ is the length of the sequence, as mentioned above. Table 1 shows the calculated mean and standard deviation (SD) values of the predicted variances $\overline{v}_i^{(s)}$ and their corresponding true values $\hat{v}_t^{(s)}$.

3.3 Experiment 3: A Stochastic Limit Cycle with State-Dependent Noise Variance

In this experiment, state-dependent noise was added to the dynamic state of the attractor, such that the noise can affect not only the current but also subsequent trajectories. The training sequences were generated by using the following piecewise stochastic differential equations.

$$\begin{cases} d\hat{y}(t) = z(t)dt + \sigma(\hat{y}(t))dB(t), \\ dz(t) = (f(z(t)) - k_1\hat{y}(t))dt. \end{cases} \tag{27}$$

$$f(z(t)) = \begin{cases} k_2 z(t) & (|z(t)| \le z_a), \\ -k_2(z(t) - 2z_a) & (z_a < z(t)), \\ -k_2(z(t) + 2z_a) & (z(t) < -z_a), \end{cases} \tag{28}$$

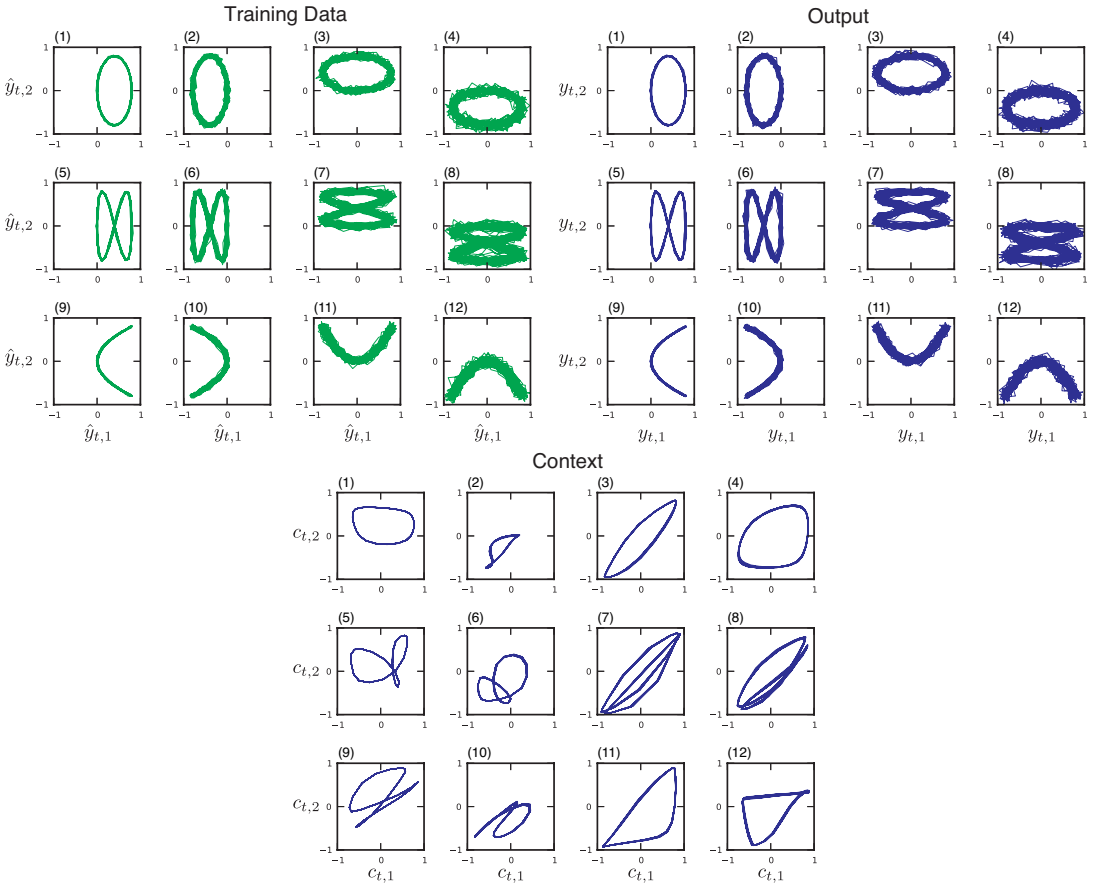

Figure 4: Comparison of phase plots. Top left: 12 Lissajous curves of training data for different values of the noise variance. Gaussian noise $\hat{\epsilon}_{t,i}^{(s)}$ is added to $\hat{y}_1$ and $\hat{y}_2$ for each pattern. In each row, the noise variance increases from left to right ($\{\hat{\sigma}^{(s)}\}^2 = 0.0001, 0.0009, 0.0025, 0.0049$). Top right: output of the trained network with closed-loop dynamics with the addition of estimated noise. Bottom: two selected context neurons of the trained network.

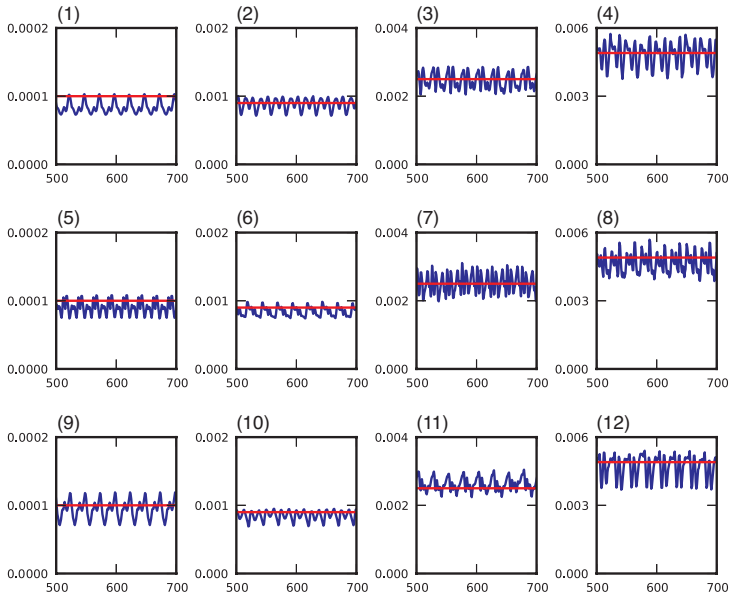

Figure 5: Time series of the predicted variances of the trained network in the case of closed-loop dynamics with addition of the estimated noise for time steps 500–700. The blue line indicates the predicted variance $v_{t,1}^{(s)}$ of the trained network, and the red line indicates its true value $\hat{v}_t^{(s)}$.

where $B(t)$ denotes Brownian motion. In fact, training sequences were computed according to the following equations, which are numerical approximations of Eqs. (27) and (28).

$$\begin{cases} \hat{y}_t = \hat{y}_{t-1} + z_{t-1}\Delta t + \hat{\epsilon}_t, \\ z_t = z_{t-1} + (f(z_{t-1}) - k_1\hat{y}_{t-1})\Delta t. \end{cases} \tag{29}$$

$$f(z_{t-1}) = \begin{cases} k_2 z_{t-1} & (|z_{t-1}| \le z_a), \\ -k_2(z_{t-1} - 2z_a) & (z_a < z_{t-1}), \\ -k_2(z_{t-1} + 2z_a) & (z_{t-1} < -z_a). \end{cases} \tag{30}$$

The parameter settings were $\Delta t = 0.1$, $k_1 = 1.0$, $k_2 = 2.0$, $z_a = 0.25$, and we defined the standard deviation $\hat{\sigma}_t$ of the added Gaussian noise $\hat{\epsilon}_t$ as follows:

$$\hat{\sigma}_t = \begin{cases} 0.03\hat{y}_{t-1} & (0 < \hat{y}_{t-1}), \\ 0 & (\text{else}). \end{cases} \tag{31}$$

In this task, we used 10 training sequences, each with a length $T = 1000$. Here, only one-dimensional sequences $\hat{y}_t$ were used for learning (z_t was used for calculation of $\hat{y}_t$).

The number of context neurons, the time constant, and the variance of the initial states were chosen to be $M = 10$, $\tau = 5$, and $\sigma^2 = 1$, respectively. We trained the network for 500,000 training steps, and the averaged MSE of all 10 sequences converged to 0.0000742. Time series were then generated with closed-loop dynamics with addition of the estimated noise by using (19b).

The result of generating time series is shown in Fig. 6. As can be seen from Fig. 6, the output of the trained network reproduces the stochastic structure of the training sequence, which contains a non-periodic pattern of the state-dependent variance. Furthermore, the predicted variance v_t of the trained network provides a close approximation of the correct value, which is calculated by $\hat{v}_t = \hat{\sigma}_t^2 = (0.03y_{t-1})^2$.

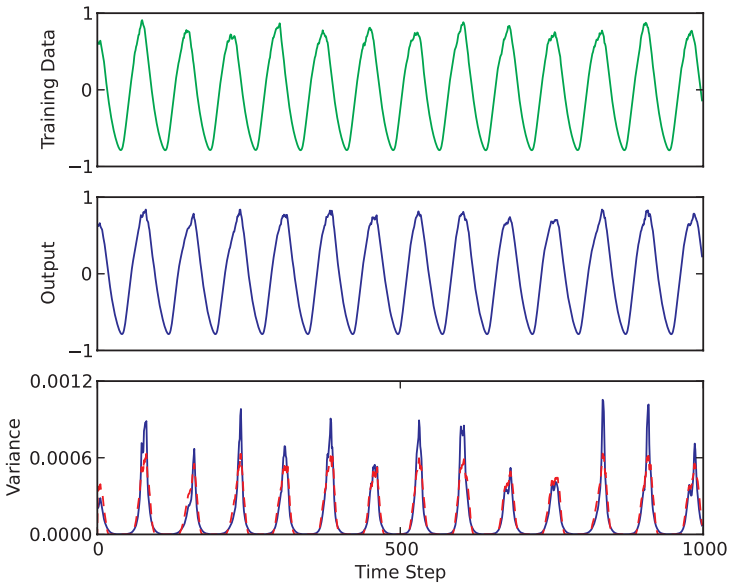

Figure 6: Generation of time series. Training sequence (first row), output of the trained network with addition of the estimated noise (second row), and variance predicted by the trained network (third row). In the third row (time series of variance), the blue line indicates the variance v_t predicted by the trained network, and the red dashed line indicates the true value $\hat{v}_t$. Here, the true value can be calculated by substituting the output of the trained network y_t into (31) as follows: $\hat{v}_{t+1} = (\hat{\sigma}_{t+1})^2 = (0.03y_t)^2$.

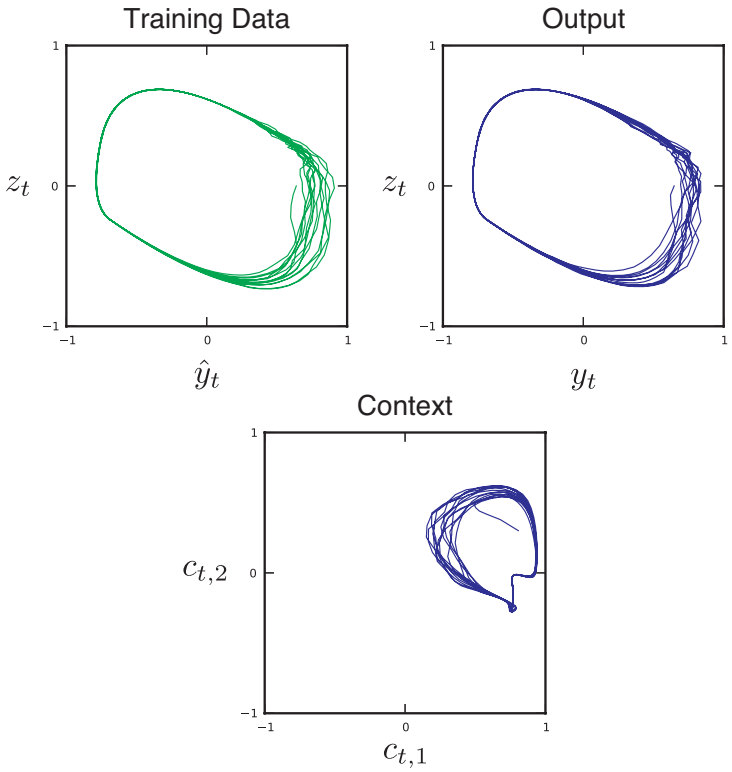

Figure 7: Comparison of phase plots. Top left: training data $\hat{y}_t - z_t$. Top right: output of the trained network with closed-loop dynamics with addition of the estimated noise $y_t - z_t$. Bottom: two selected context neurons of the trained network.

Table 1: Comparison between Mean of Predicted Variances and the Corresponding True Values.

Index of Sequence	**True Value**	**Mean and SD of Predicted Variance**			
s	$\hat{v}^{(s)}$	$\overline{v}_1^{(s)}$	$\mathrm{SD}_1^{(s)}$	$\overline{v}_2^{(s)}$	$\mathrm{SD}_2^{(s)}$
1	0.0001	0.000084	0.0000090	0.000080	0.0000058
2	0.0009	0.000874	0.0000919	0.000901	0.0000827
3	0.0025	0.002452	0.0002574	0.002414	0.0002623
4	0.0049	0.004813	0.0005368	0.004799	0.0005712
5	0.0001	0.000090	0.0000094	0.000086	0.0000101
6	0.0009	0.000837	0.0000717	0.000854	0.0000912
7	0.0025	0.002525	0.0002705	0.002448	0.0001840
8	0.0049	0.004602	0.0004271	0.004654	0.0003304
9	0.0001	0.000094	0.0000130	0.000094	0.0000090
10	0.0009	0.000846	0.0000679	0.000854	0.0000700
11	0.0025	0.002614	0.0002061	0.002521	0.0002025
12	0.0049	0.004581	0.0005425	0.004545	0.0004428

Figure 7 presents phase plots of the training data $\hat{y}_t - z_t$, the output of the trained network $y_t - z_t$, and the two selected context neurons. Comparing the phase plots of the training data and the closed-loop dynamics, we can see that the trained network can generate a stochastic dynamic sequence similar to the training data. Furthermore, it appears that the context state also shares the same stochastic dynamic structure.

4 Robot Experiment

We also conducted a robot experiment involving training and reproduction of sensory-guided stochastic behavior in order to evaluate the performance of the S-CTRNN model in more realistic situations, in which it is difficult to identify possible sources of noise and the characteristics of the noise.

4.1 Obtaining Training Sequences

In this experiment, a trainer was instructed to repeatedly demonstrate visually guided movement patterns (reaching toward a visually perceived object) to a small humanoid robot called "NAO" by physically guiding its arm. The robot was seated on the floor facing a workbench. In generating the training patterns, the trainer was instructed to guide the robot's hand precisely

until it touched a red sphere affixed to the workbench and to subsequently retract the hand without following a precise trajectory, as illustrated in Fig. 8. The red sphere could be located at

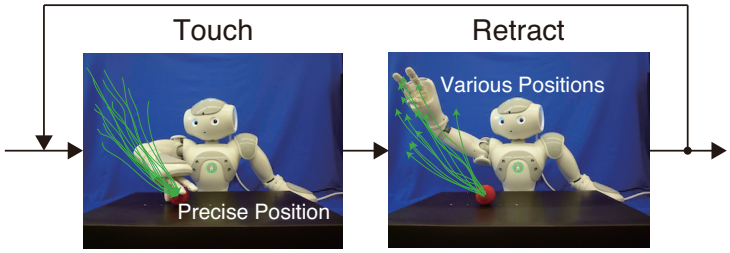

Figure 8: Movement sequence learned during the training session. The task for the robot was simply to touch a red sphere placed on the workbench with its right hand and then retract its hand. This touch-and-retract action was repeated several times to obtain a training sequence that included various trajectories.

different positions with respect to the plane of symmetry between the left and right sides of the robot. What can be expected after successfully training the S-CTRNN with fluctuating training trajectories demonstrated by a human trainer is that the closed-loop forward dynamics with an accurate estimate of the mean values and their variances at each time step, can reproduce the trajectories learned through visual guidance by inferring the stochastic structure hidden in the training sequences.

The joint angles of the robot's head were controlled to fixate automatically on the center of the target object, regardless of the robot's actions both while it was being guided by the trainer to obtain training data and in generating actions using the trained network. Therefore, the direction of the head provided visual information about the relative location of the object in this experiment. Reading the two joint angles of the head (yaw and pitch) provided information about the visual target (position of the red sphere), and reading the joint angles of the right arm (shoulder pitch, shoulder roll, elbow yaw, and elbow roll) provided information about the proprioceptive inputs. The angles of the remaining joints were fixed. During the process of guiding the robot's hand, the changes in the joint angles of the head and right arm were recorded. These joint angles were mapped to values ranging between -0.8 and 0.8 because the activation function of the output used for S-CTRNN was tanh (ranging between -1.0 and 1.0).

The object was located at one of three positions (denoted by Positions 1–3) (Fig. 9) in the tutoring phase, and it was affixed at the same position while the trainer repeatedly demonstrated the touch-and-retract action. Here, Position 3 was the center of the workbench on the line of

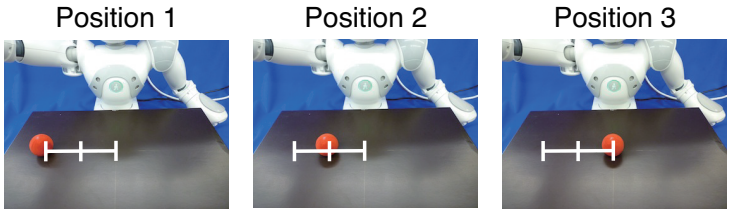

Figure 9: Three positions of the object. The distance between two neighboring positions was 7 cm.

symmetry between the left and right sides of the robot.

The touch-and-retract action guided by the trainer was repeated 15 times for one training sequence. Note that the touch-and-retract action in each training sequence was not always completely periodic, although the trainer attempted to generate precise periodic patterns by

using a metronome. The actual learning took place with such noisy data. To ensure robust learning, 10 training sequences were recorded for each of the 3 object positions. Therefore, the total number of training sequences used in the learning phase was 30, for a total of 450 touch-and-retract actions. The training sequences were recorded every half a second, and the length of each sequence was about $L = 180$. After the period of training the network, the robot was tested for reproduction of the learned visually guided actions with the object located in learned as well as in new positions.

4.2 System Architecture

Figure 10 presents an overview of the system architecture for the generation mode. Inputs to the

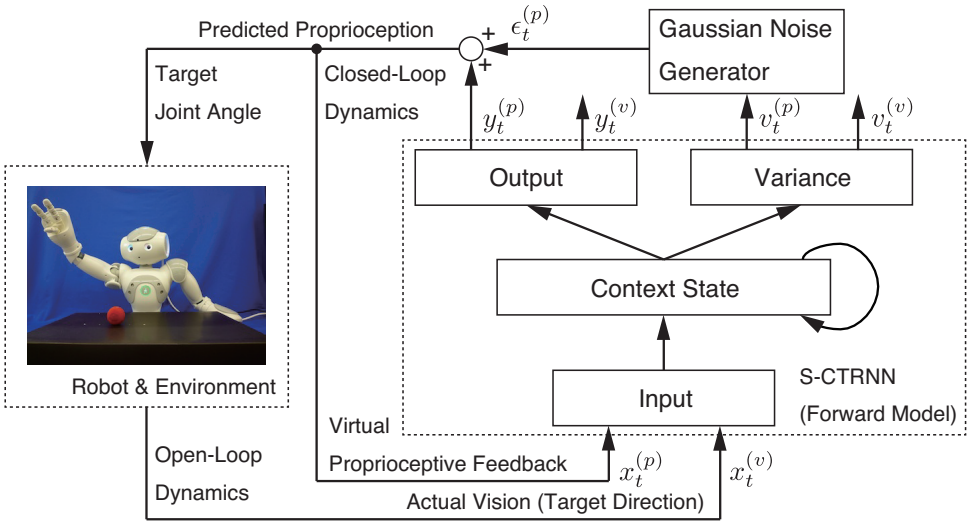

Figure 10: System architecture for the generation mode. The S-CTRNN model was used as a forward model for controlling the robot. Inputs to the network were in the form of visuo-proprioceptive sequences $x_t^{(v)}$ and $x_t^{(p)}$. The network was trained to predict the inputs for the next time step. The predicted proprioceptive sequence $y_t^{(p)}$ with addition of the estimated noise $\epsilon_t^{(p)}$ corresponding to the predicted variances $v_t^{(p)}$ was sent to the robot in the form of target joint angles, which acted as motor commands for the robot to generate movements. By adding Gaussian noise, the network was able to reproduce the stochastic dynamical structure of the training sequences.

network were in the form of visuo-proprioceptive sequences $x_t^{(v)}$ and $x_t^{(p)}$. The network predicted the input state for the next time step as output sequences $y_t^{(v)}$ and $y_t^{(p)}$, and also predicted the variances $v_t^{(v)}$ and $v_t^{(p)}$ corresponding to each output. The predicted proprioceptive sequences $y_t^{(p)}$ with addition of the estimated noise $\epsilon_t^{(p)}$ corresponding to the predicted variances $v_t^{(p)}$ was fed into the next input state, and network output values were remapped into joint angles. The remapped values were sent to the robot in the form of target joint angles, which acted as motor commands for the robot to generate movements. However, the predicted visual sequence $y_t^{(v)}$ and variance $v_t^{(v)}$ were not used for action generation because the joint angles of the robot's head were controlled independently, as mentioned above. Therefore, the inputs to the network

were expressed as follows:

$$\begin{cases} x_t^{(p)} = y_{t-1}^{(p)} + \epsilon_{t-1}^{(p)}, \\ x_t^{(v)} = \hat{y}_{t-1}^{(v)}. \end{cases} \tag{32}$$

For action generation, initial states $u_{0,i}^{(s)}$ corresponding to each sequence s were not used, and instead the mean value $\hat{u}_i$ of the initial states of all training sequences was used. Therefore, differences in behavior trajectories are not due to the initial state of the trained network but arise from differences in the visuo-proprioceptive sequences (i.e., location of the object) and the added Gaussian noise.

4.3 Results

The number of context neurons, the time constant, and the variance of the initial states were chosen to be $M = 30$, $\tau = 2$, and $\sigma^2 = 0.001$, respectively. We trained the network for 500,000 training steps, and the averaged MSE of all 30 sequences converged to 0.000763. Action sequences performed by the robot were then generated by the trained network.

Figure 11 illustrates an example of visuo-proprioceptive sequences in the training data, output sequences and variance generated by the trained network. As can be seen from Fig.

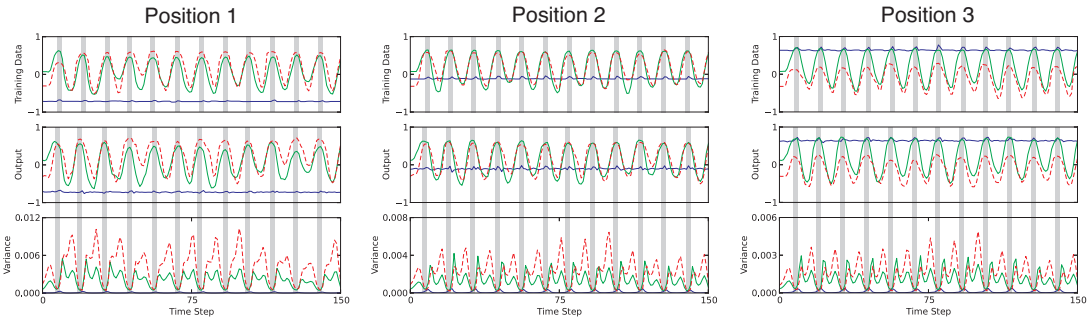

Figure 11: Generation of time series for each object position (Positions 1–3). Training sequence (first row), output of the trained network with closed-loop dynamics with addition of the estimated noise (second row), and predicted variance of the trained network with closed-loop dynamics with addition of the estimated noise (third row). Each panel shows a plot of 1 out of 2 vision dimensions (blue: head pitch) and 2 out of 4 proprioception dimensions (green: right shoulder pitch, red dashed: right elbow roll). The upper parts of the trajectories of the training data and the output correspond to movement directed towards touching the object, and the lower parts correspond to retraction of the robot's arm. The gray areas correspond to the moments at which the robot touched the object.

11, the output of the trained network with closed-loop dynamics with addition of the estimated noise reproduces the stochastic structure of the training sequence, which contains cyclic patterns with stochastic variances. Furthermore, the regions of increase and decrease can be observed in the predicted variances, where we see that the variance decreases to the minimum at the very moment of touching the object (at the crests highlighted in gray) and increases after the robot retracts its arm (at the intermittent valleys).

Figure 12 shows snapshots capturing the moments of touching and after retracting for each object position. As seen in Fig. 12, the position of the robot's hand is centered at the point corresponding to the position of the object. However, it varies when the robot retracts its hand after touching the object.

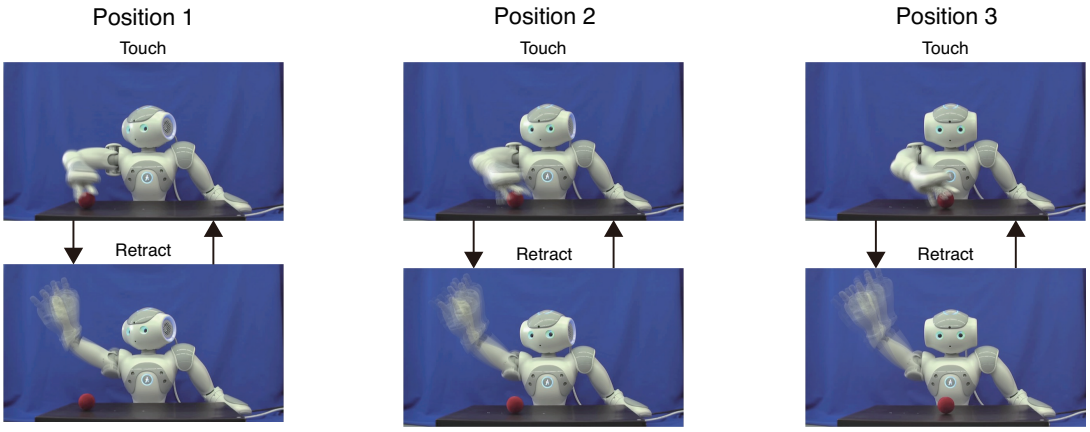

Figure 12: Snapshots of action sequences performed by NAO controlled by the trained network for each object position (Positions 1–3). The upper panels correspond to movement directed towards touching the object, and the lower panels correspond to the moment after the robot retracts its arm.

The robot equipped with the trained network was able to reproduce the corresponding stochastic behavior not only in the cases where the object was located at the learned positions, but also when it was placed in new positions which did not appear in the training data, for example, between Positions 1 and 2 and between Positions 2 and 3, by means of generalization. Moreover, the robot was able to adapt to perturbations such as sudden changes in the object location in the action generation stage, although the robot had never experienced such situations in the learning process. Specifically, as long as the object was within the range between Positions 1 and 3, including new positions, the robot was able to touch the object even if the object was shifted from its original position when the robot started to move.

Figure 13 presents phase plots of the training sequence, the output of the trained network and the two selected context neurons of the trained network for each position, including cases where the object is located between learned positions. In comparing Fig. 13, the phase plots of the training sequence and those of the output of the trained network share similar stochastic dynamical structures in which the trajectories tend to converge in the upper right corner of the phase plot for the shoulder and the upper left corner of the phase plot for the elbow (corresponding to the moment of touching the object), and diverge in the lower left corner of the phase plot for the shoulder and the lower right corner of the phase plot for the elbow (corresponding to the moment of retracting the arm). Furthermore, the same stochastic properties can be seen in context states. Additionally, the trajectories of output and context states for new positions appear to be situated between those of the two learned positions and show the same stochastic properties.

These results suggest that the proposed S-CTRNN model can reproduce the observed fluctuating trajectories to some extents by inferring hidden stochastic structures in terms of time-dependent means and variances of the network outputs. Furthermore, the generalization capabilities via learning for new sensory inputs (e.g., new object locations) were also confirmed.

5 Discussion

In the numerical experiments with various values for the variance of Gaussian noise, the proposed S-CTRNN was able to reproduce the stochastic properties hidden in the training data by estimating the variance correctly. In the first experiment, described in Section 3.1, we con-

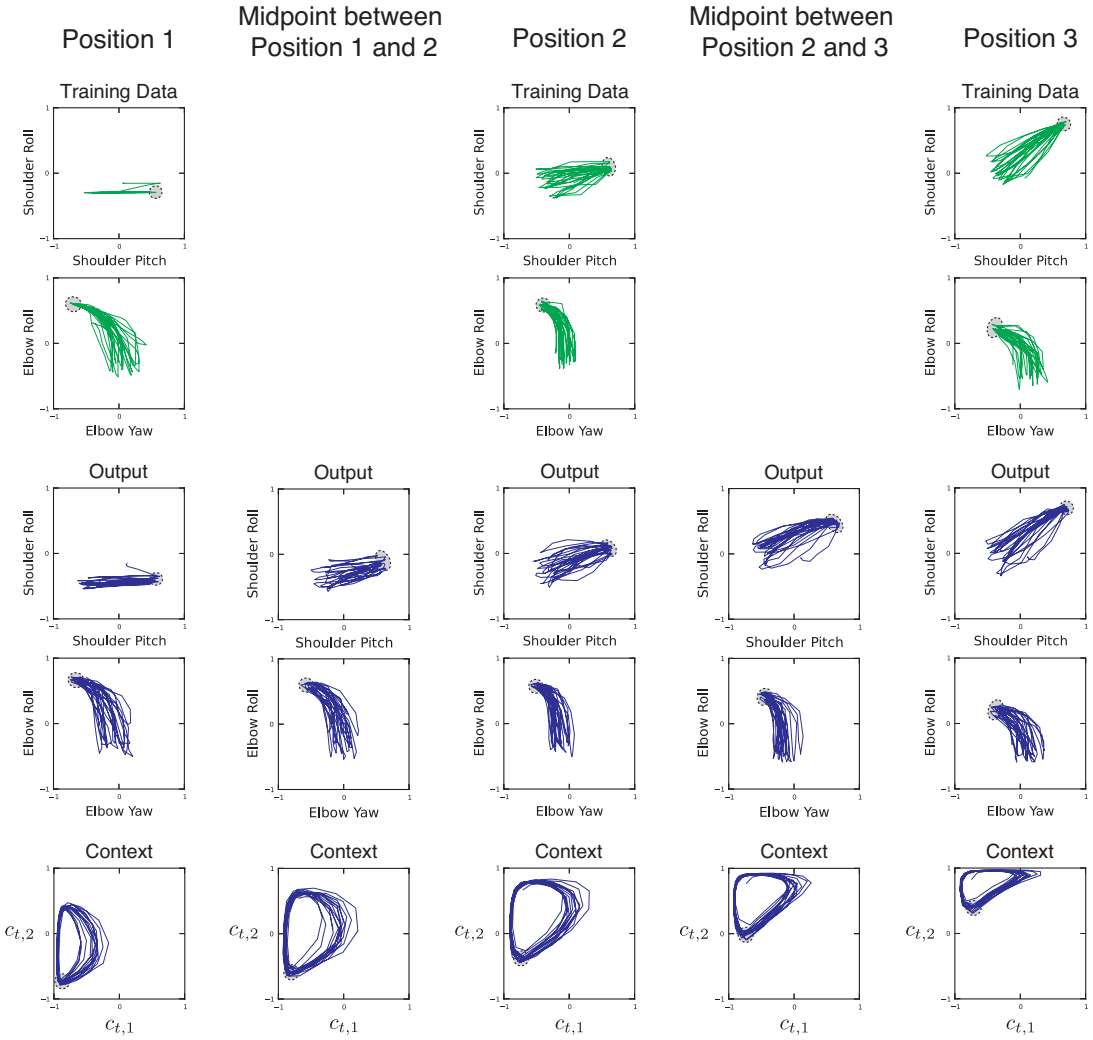

Figure 13: Phase plots of the training data, the output of the trained network, and the two selected context neurons of the trained network for each object position. Shoulder space of the training data (first row), elbow space of the training data (second row), shoulder space of the output (third row), elbow space of the output (fourth row), and context space (fifth row). The gray areas correspond to the moments at which the robot touched the object.

sidered a simple task in which the training sequence was a one-dimensional sinusoidal curve with added Gaussian noise with a time-dependent variance in order to compare the variance estimated by the trained network and that in the training sequence. In Fig. 3, we can see that the estimated variance is close to the target one. From this result, it was concluded that the proposed S-CTRNN can predict the time-dependent noise variance added to the reference trajectory. In the second experiment, described in Section 3.2, the task involved learning multiple Lissajous curve patterns with different shapes with added Gaussian noises with different values of the constant variance. The experimental results revealed that the training patterns can be reproduced as multiple limit cycle attractors with variance which has been adequately estimated for each pattern through the learning process. It should be noted that in both of the abovementioned experiments, Gaussian noise was added not to the intrinsic dynamics of generating those patterns but directly to the reference trajectories of the training data at each time step. In contrast, in the third experiment (described in Section 3.3), we added Gaussian noise to the dynamics of the limit cycle attractor itself. In this case as well, the network was able to predict the variance correctly (see Fig. 6) and to reproduce the stochastic structure hidden in the training data (see the training data and the output in Fig. 7). By comparing the context states in Figs. 4 and 7, we can see a qualitative difference between their trajectories. In the case where noise was added to the reference trajectories, such as in the experiments in Sections 3.1 and 3.2, future states are independent of any noise added at previous steps. Therefore, the hidden stochastic structures did not appear in the context state trajectories of the trained network (see context states in Fig. 4). On the other hand, in the case where noise was added to the dynamic system generating the training data, such as in the case described in Section 3.3, stochastic structures appeared in the context trajectories in the trained network, such that noise added at previous steps affected future states (see context states in Fig. 7).

In the robot experiment, S-CTRNN was also able to reproduce fluctuating trajectories demonstrated by human trainers. The robot controlled by the trained network reproduced the sensor-guided behavior and exhibited fluctuations with a structure similar to that of the training trajectories. Although it is difficult to identify sources of noise or to estimate the characteristics of noise directly in this real-world situation, at least it was observed that the network was capable of reproducing fluctuating trajectories with a structure similar to that of the tutored trajectories. Such behavioral trajectories were generated successfully even for new sensory input through generalization of learning. Moreover, the robot was able to adapt to perturbations such as sudden changes in the object position during action generation. These results suggest that the proposed model can achieve generalization and adaptation to some extent by learning.

These numerical and robot experiments indicate that the proposed mechanism of estimating variance shows certain advantages in the process of learning and reproduction of target stochastic sequences. In particular, it should be noted that the proposed learning scheme based on estimating variance in training sequences can facilitate the learning processes because the weight of error signals for noisy sequences is reduced before back-propagation as it is divided by the variance. Our preliminary investigation suggested that most of the present results can be regenerated robustly for a variance of up to 30% in the presented parameters, including the number of context neurons, the time constant and the learning rate. The crucial part of the parameter setting is that the learning rate should be scaled by the total length of the training sequences and the dimensionality of the output neurons as described in Section 2.4 because the maximum value of the likelihood depends largely on them.

Several issues remain to be examined in future studies. One such issue is the "temporal registration" problem. In the probabilistic approach employing a combination of GMM and GMR [14–16], dynamic time warping (DTW) is used for temporal normalization because GMM

contains time as a variable, and the mean and variance information for a given time step is calculated with GMR. For performing expert-level aerobatics by an autonomous helicopter, Coates et al. [22] proposed an algorithm that extracts the unobserved target trajectory that an expert pilot was trying to demonstrate from multiple suboptimal demonstrations, and then constructs an accurate dynamics model by using the extracted trajectory for controlling the helicopter. In the former process, they also used DTW to enable the model to be simpler. In the current study, temporal registration was not applied to the training sequence because there is no time warping in the data. Although it can be argued that the context states can absorb time warping to a certain extent as well as other dynamical systems approaches [23, 24], the acceptable range should be investigated in future work.

Another issue concerns the utilization of the predicted variance for the robot's action generation. For example, when the humanoid robot ASIMO was given a pouring task in [16], the learned variance information was interpreted as a measure of the importance of parts of the pouring movement. In parts with high variance, the robot's movement diverged more from the actual learned movement in the case of avoiding self-collision because the required movement precision in this part of the task execution was lower. Although the predicted variance was used only to reproduce stochastic structures in the training data in the current study, it is considered that our proposed model can also be applied to the acquisition of skilled behavior by mediating physical constraints and the variability in generating trajectories, as demonstrated in the aforementioned example.

Finally, the last issue concerns the learning capabilities of the S-CTRNN. In numerical experiments, the network was able to estimate both the time-dependent noise variance added to the reference trajectory and the noise variance added to the intrinsic dynamics. For evaluating the learning capabilities in similar setups, we conducted additional numerical experiments in which the standard deviation of the added Gaussian noise was three times larger than that in each numerical experiment described in Section 3. Consequently, we found similar results in the additional experiments. In the robot experiment, the same network was able to reproduce fluctuating trajectories with structures similar to those of the observed trajectories. The current model of S-CTRNN predicts mean of $\boldsymbol{x}(t + \Delta t)$ and its variance with $\boldsymbol{x}(t)$ given, where $\boldsymbol{x}(t)$ and $\boldsymbol{x}(t+\Delta t)$ are input state (e.g., visuo-proprioceptive state for the robot experiment) at time step t and $t + \Delta t$, respectively. If Δt is sufficiently small, the mapping function from $\boldsymbol{x}(t)$ to $\boldsymbol{x}(t + \Delta t)$ can be approximated as a linear one by which its likelihood should have a single peak normal distribution. Therefore the assumption of normal distribution should work for one step prediction with small Δt. If Δt becomes large, the assumption of normal distribution for the likelihood cannot be held because of the nonlinearity in the prediction mapping from t to $t + \Delta t$. The assumption is not a general limitation of the proposed approach, since it can be extended to other probability distributions. We demonstrated only one successful example of reproducing fluctuating structures via observation, and therefore future studies should evaluate the availability and limitations of the proposed scheme in more diverse cases.

6 Conclusion

In this study, we presented results indicating that the proposed S-CTRNN model can learn and reproduce both artificial and human-made stochastic time series by estimating both the time-dependent mean and the variance of observed trajectories. In order to evaluate the capabilities of the S-CTRNN model, we conducted three experiments with various conditions for the fluctuating trajectories. In all three experiments, S-CTRNN was able to learn and reproduce trajectories by extracting the stochastic structures hidden in the target training trajectories. Furthermore, the results of the robot learning experiment suggest that the proposed scheme

can be applied to real-world problems, including teaching skilled behavior to humanoid robots. Future studies should evaluate the possible scope of application of the proposed scheme.

Appendix Training Data: Stochastic Lissajous Curves with Multiple Constant Values for the Noise Variance

The equations for the Lissajous curves in training data in Section 3.2 are as follows

$$(\hat{y}^{(1)}_{t,1}, \hat{y}^{(1)}_{t,2}) = (Y_{t,1} + \hat{\epsilon}^{(1)}_{t,1}, Y_{t,2} + \hat{\epsilon}^{(1)}_{t,2}), \tag{33}$$
$$(\hat{y}^{(2)}_{t,1}, \hat{y}^{(2)}_{t,2}) = (-Y_{t,1} + \hat{\epsilon}^{(2)}_{t,1}, Y_{t,2} + \hat{\epsilon}^{(2)}_{t,2}), \tag{34}$$
$$(\hat{y}^{(3)}_{t,1}, \hat{y}^{(3)}_{t,2}) = (Y_{t,2} + \hat{\epsilon}^{(3)}_{t,1}, Y_{t,1} + \hat{\epsilon}^{(3)}_{t,2}), \tag{35}$$
$$(\hat{y}^{(4)}_{t,1}, \hat{y}^{(4)}_{t,2}) = (Y_{t,2} + \hat{\epsilon}^{(4)}_{t,1}, -Y_{t,1} + \hat{\epsilon}^{(4)}_{t,2}), \tag{36}$$
$$(\hat{y}^{(5)}_{t,1}, \hat{y}^{(5)}_{t,2}) = (Y_{t,1} + \hat{\epsilon}^{(5)}_{t,1}, Y_{t,4} + \hat{\epsilon}^{(5)}_{t,2}), \tag{37}$$
$$(\hat{y}^{(6)}_{t,1}, \hat{y}^{(6)}_{t,2}) = (-Y_{t,1} + \hat{\epsilon}^{(6)}_{t,1}, Y_{t,4} + \hat{\epsilon}^{(6)}_{t,2}), \tag{38}$$
$$(\hat{y}^{(7)}_{t,1}, \hat{y}^{(7)}_{t,2}) = (Y_{t,4} + \hat{\epsilon}^{(7)}_{t,1}, Y_{t,1} + \hat{\epsilon}^{(7)}_{t,2}), \tag{39}$$
$$(\hat{y}^{(8)}_{t,1}, \hat{y}^{(8)}_{t,2}) = (Y_{t,4} + \hat{\epsilon}^{(8)}_{t,1}, -Y_{t,1} + \hat{\epsilon}^{(8)}_{t,2}), \tag{40}$$
$$(\hat{y}^{(9)}_{t,1}, \hat{y}^{(9)}_{t,2}) = (Y_{t,3} + \hat{\epsilon}^{(9)}_{t,1}, Y_{t,2} + \hat{\epsilon}^{(9)}_{t,2}), \tag{41}$$
$$(\hat{y}^{(10)}_{t,1}, \hat{y}^{(10)}_{t,2}) = (-Y_{t,3} + \hat{\epsilon}^{(10)}_{t,1}, Y_{t,2} + \hat{\epsilon}^{(10)}_{t,2}), \tag{42}$$
$$(\hat{y}^{(11)}_{t,1}, \hat{y}^{(11)}_{t,2}) = (Y_{t,2} + \hat{\epsilon}^{(11)}_{t,1}, Y_{t,3} + \hat{\epsilon}^{(11)}_{t,2}), \tag{43}$$
$$(\hat{y}^{(12)}_{t,1}, \hat{y}^{(12)}_{t,2}) = (Y_{t,2} + \hat{\epsilon}^{(12)}_{t,1}, -Y_{t,3} + \hat{\epsilon}^{(12)}_{t,2}). \tag{44}$$

References

[1] Eleanor J. Gibson and Pick Anne D. *An Ecological Approach to Perceptual Learning and Development*. Oxford University Press, 2000.

[2] D M Wolpert and M Kawato. Multiple paired forward and inverse models for motor control. *Neural Netw*, 11(7-8):1317–29, 1998.

[3] Martin V. Butz, Olivier Sigaud, Giovanni Pezzulo, and Gianluca Baldassarre, editors. *Anticipatory Behavior in Adaptive Learning Systems: From Brains to Individual and Social Behavior, LNAI 4520 (State-of-the-Art Survey).* Springer-Verlag, Berlin Heidelberg, 2007.

[4] Michael I Jordan. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16:307–354, 1992.

[5] Jeffrey L Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[6] Ronald J Williams and David Zisper. A learning algorithm for continually running fully recurrent neural network. *Neural Computation*, 1(2):270–280, 1989.

[7] Jun Tani. Model-based learning for mobile robot navigation from the dynamical systems perspective. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(3):421–436, 1996.

[8] Masato Ito, Kuniaki Noda, Yukiko Hoshino, and Jun Tani. Dynamic and interactive generation of object handling behaviors by a small humanoid robot using a dynamic neural network model. *Neural Networks*, 19:323–337, 2006.

[9] Yuichi Yamashita and Jun Tani. Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS Comput Biol*, 4(11):e1000220, 2008.

[10] Shun Nishide, Jun Tani, Toru Takahashi, Hiroshi G. Okuno, and Tetsuya Ogata. Tool–body assimilation of humanoid robot using a neurodynamical system. *IEEE Transactions on Autonomous Mental Development*, 4(2):139–149, 2012.

[11] Jun Namikawa, Ryunosuke Nishimoto, Hiroaki Arie, and Jun Tani. Synthetic approach to understanding meta-level cognition of predictability in generating cooperative behavior. *Proceedings of the Third International Conference on Cognitive Neurodynamics*, in press.

[12] Karl Friston. The free-energy principle: a rough guide to the brain? *Trends Cogn Sci*, 13(7):293–301, 2009.

[13] Karl Friston, Jérémie Mattout, and James Kilner. Action understanding and active inference. *Biol Cybern*, 104(1-2):137–60, 2011.

[14] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 37:2:286–298, 2007.

[15] Sylvain Calinon and Aude Billard. Statistical learning by imitation of competing constraints in joint space and task space. *Advanced Robotics*, 23:15:2059–2076, 2009.

[16] Manuel Muehlig, Michael Gienger, Sven Hellbach, Jochen J. Steil, and Christian Goerick. Task-level imitation learning using variance-based movement optimization. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA 2009)*, pages 1177–84, 2009.

[17] Kenji Doya and Yoshizawa Shuji. Adaptive neural oscillator using continuous-time back-propagation learning. *Neural Netw*, 2:375–385, 1989.

[18] Kenichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Netw*, 6:801–806, 1993.

[19] Randall D. Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(3):469–509, 1995.

[20] Jun Namikawa and Jun Tani. A model for learning to segment temporal sequences, utilizing a mixture of rnn experts together with adaptive variance. *Neural Netw*, 21(10):1466–75, 2008.

[21] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. In David E Rumelhart and David McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge, MA, 1986.

[22] Adam Coates, Pieter Abbeel, and Andrew Y. Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th international conference on Machine learning*, pages 144–151, 2008.

[23] Elena Gribovskaya, S. Mohammad Khansari-Zadeh, and Aude Billard. Learning nonlinear multivariate dynamics of motion in robotic manipulators. *International Journal of Robotics Research*, 30(1):80–117, 2011.

[24] S. Mohammad Khansari-Zadeh and Aude Billard. Learning stable non-linear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.