Predictive Learning with Uncertainty Estimation for Achieving Adaptive and Flexible Behavior

February 2016

Shingo MURATA

Doctoral Thesis

Predictive Learning with Uncertainty Estimation for Achieving Adaptive and Flexible Behavior

February 2016

Waseda University

Graduate School of Creative Science and Engineering

Department of Modern Mechanical Engineering,
Research on Intelligent Machine

Shingo MURATA

Abstract

One of the most fascinating human cognitive abilities is adaptive and flexible behavior generation together with adequate action, perception, and attention in dynamic and uncertain environment. Such an ability must also be necessary for artificial agents, for example, intelligent robots that are expected to provide livelihood support in everyday life. Therefore, acquisition of the ability to generate adaptive and flexible behavior is a crucial issue for these robots.

Looking at the developmental process of human infants, they gradually acquire cognitive abilities through dynamic interactions with their physical and social environment. In particular, at the early stage of the developmental process, interactions with their caregivers who actively support them are essential to achieve cognitive behavior such as playing with objects, imitation, and joint attention. They further acquire more social cognitive skills through playing with other infants or children who may sometimes violate their expectations about others' behavior. What human infants learn through these interactions is what will happen when they do something, namely, the relationship between causal states and their consequences. This can be cast as predictive learning of sensory consequences including exteroceptive (or visual) and proprioceptive states which are caused by the self and the external world including others. The predictive learning is enabled by so-called internal or generative models of the world which are considered to be acquired in the brain. A study on acquisition of generative models is crucial for both understanding brain mechanisms of biological agents and implementing such mechanisms in artificial agents.

Recurrent neural networks (RNNs) have been adopted as one of possible computational frameworks to specify these models thanks to their input- and context-dependent predictive learning capabilities under the simple computational principle of prediction error minimization. RNNs can learn to generate predictions about the next state of target temporal sequence data by receiving the current state of those as input and incorporating contextual dynamics stored in the networks. This thesis explores a computational framework of RNNs that enables intelligent robots to achieve adaptive and flexible behavior which can be performed in real environment.

Conventional RNN-based frameworks, however, face three potential problems or issues due to their deterministic properties by which all data structures are modeled as deterministic dynamical systems even though the data are the successive states of stochastic processes. The first is that if RNNs are forced to learn multiple temporal sequence data with stochastic or random fluctuations, the learning process tends to become unstable with the accumulation of errors. The second is their inability to learn to extract and reproduce stochastic structures, such as the amplitude of the fluctuations, latent in the data. These two issues regarding fluctuations or uncertainty at a trajectory level are tightly coupled and important to consider the acquisition of adaptive action primitives. The third is that RNNs cannot estimate uncertainty at an event level such as transition probability of the action primitives. The estimation of event-level uncertainty is important to consider the development of reactive and proactive behavior for realizing flexible behavior that produces various action sequences consisting of the action primitives.

The aim of this thesis is to develop an advanced RNN-based framework for intelligent robots to acquire generative models of the external world for achieving adaptive and flexible behavior by tackling these three non-trivial issues regarding the different levels of uncertainty. The framework enables RNNs to learn to predict both the mean and the variance of the next state of target data with fluctuations, where the variance corresponds to the uncertainty of target variables and the reciprocal of the variance is called precision. Studies described in this thesis especially consider the implementation of this framework on conventional continuous-time RNNs (CTRNNs) whose context states employ leaky-integrator neural units to deal with continuous data flow. The novel network is referred to as stochastic CTRNN (S-CTRNN) in terms that an aspect of stochasticity is introduced into the conventional CTRNN. The additional variance prediction mechanism enables the S-CTRNN (1) to stably learn multiple temporal sequence data with random fluctuations because the learning process tries to minimize prediction errors scaled by the predicted variance, which are called precision-weighted prediction errors, and (2) to extract and reproduce stochastic structures latent in such data in terms of the time-varying mean and variance states. These two points contribute to solve the aforementioned first and second issues. For the third issue, an extension of the S-CTRNN is considered. The extended hierarchical network referred to as stochastic multiple timescale RNN (S-MTRNN) consists of the lower-level and higher-level subnetworks with different timescale

dynamics of the context states. The S-MTRNN can self-organize internal representations of primitives in the lower-level subnetwork with fast dynamics and those of sequences of the primitives in the higher-level subnetwork with slow dynamics. By incorporating this ability to self-organize hierarchical representations and the variance prediction mechanism, the S-MTRNN is able (3) to determine its modeling way, namely, probabilistic or deterministic modeling, depending on the condition of learning.

The abilities of these stochastic RNNs are verified through a series of numerical experiments and two kinds of robot experiments. The numerical experiments in which artificial training data are generated in different ways utilizing Gaussian noise consider the first and second issues regarding the trajectory-level uncertainty. The first robot experiment on learning reaching movement toward a fixed target object also considers these issues in more realistic situations. The results demonstrate the development of adaptive behavior by estimating the trajectory-level uncertainty. The second robot experiment on learning interactive behavior with another robot considers the third issue regarding the event-level uncertainty. The results demonstrate the development of flexible behavior based on reactive and proactive behavior by estimating the event-level uncertainty.

This thesis consists of six chapters. Chapter 1 provides the background, research questions, hypotheses, research objective, related work, and overview of the proposed approach as an introduction of the current study.

Chapter 2 proposes a novel computational framework that enables RNNs to deal with stochasticity. The form of generative model is first introduced and then an architecture based on the CTRNN is introduced. Computational methods of forward propagation for generating predictions and backward propagation for the learning are provided. After specifying the basic architecture of the S-CTRNN, an advanced architecture called S-MTRNN is introduced together with a novel dynamic recognition method called error regression scheme (ERS) which modifies higher-level internal neural dynamics online by minimizing precision-weighted prediction errors.

Chapter 3 demonstrates the results of a series of numerical experiments on learning and reproduction of temporal sequence data with random fluctuations. The first experiment demonstrates the comparison of learning capabilities between the conventional CTRNN and the proposed S-CTRNN by using a set of temporal sequence data with random fluctuations. The second experiment demonstrates the learning results of a sinusoidal curve with random fluctuations

whose variance changes depending on the time. The third experiment demonstrates the learning results of random dynamical system in which the variance for the added Gaussian noise changes depending on the state of the system itself. Through these numerical experiments, the ability of the S-CTRNN for extracting different types of uncertainty is demonstrated.

Chapter 4 demonstrates the results of a robot experiment on learning to develop adaptive behavior from human demonstrations via kinesthetic teaching. The S-CTRNN is applied to robot learning in the context of primitive skill acquisition for reaching movement toward a target object. The human demonstrations include stochastic structures such that some parts are variant and other parts are invariant. The robot equipped with the S-CTRNN is required to learn these structures specifying a task constraint. After the learning process, the robot is able to generate adaptive behavior together with the stochastic structures not only in learned situations but also in unlearned situations by utilizing generalization ability of the S-CTRNN.

Chapter 5 demonstrates the results of a robot experiment on learning to develop flexible behavior. The S-MTRNN is applied to robot learning in the context of learning to interact with another agent. The experimental results show that self-organized reactive behavior–based on probabilistic prediction with high event-level uncertainty–emerges when learning proceeds without a precise specification of initial conditions of the S-MTRNN. In contrast, proactive behavior with deterministic predictions with low event-level uncertainty emerges when precise initial conditions are available. These results indicate that two different ways of treating uncertainty about perceptual events in learning, namely, probabilistic modeling and deterministic modeling, contribute to the development of different dynamic neuronal structures governing the two types of behavior generation schemes.

Chapter 6 summarizes the achievements of a series of both numerical and robot experiments. The reviews on remaining issues and on future directions conclude this thesis.

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. Shigeki Sugano, for providing me this precious study opportunity as a doctoral student in his laboratory. Over the last six years including studies for the bachelor's and master's degrees, his objective and logical advices on both my research and life as a researcher have supported me many times. I would also like to thank my four examiners, Prof. Masakatsu G. Fujie, Prof. Tetsuya Ogata, Prof. Tomoyuki Miyashita, and Prof. Hiroyasu Iwata, for providing me accurate and helpful comments on my presentation and an earlier version of this thesis. Especially, I am very grateful to Prof. Ogata for allowing me to be a member of his laboratory from the first year of the doctoral course and for providing me insightful comments in weekly group meetings.

I would like to thank Prof. Jun Tani of Korea Advanced Institute of Science and Technology (KAIST). He has taught me a lot of things for becoming a good researcher since he was a team leader of the Lab. for Behavior and Dynamic Cognition (BDC), RIKEN Brain Science Institute, where I was a trainee from the fourth-year undergraduate to the beginning of the second-year master student. I couldn't have concluded this research without his kind guidance and support for six years. I have been greatly influenced by his attitude as a researcher from which everyone can understand that he does enjoy his research. I am very grateful to all the BDC Lab. ex-members, especially Dr. Jun Namikawa and Dr. Yuichi Yamashita for their technical supports and helpful discussions.

I would like to thank Dr. Hiroaki Arie, Dr. Kuniaki Noda, and Dr. Yuki Suga for their valuable comments on my study. I would also like to thank the graduate and undergraduate students belonging to "Cognitive Robotics" group, a joint research group between Sugano Lab. and Ogata Lab., especially, Ms. Yiwen Chen, Ms. Yuxi Li, and Ms. Saki Tomioka from Sugano Lab.; and Ms. Kai Hirano, Mr. Tatsuro Yamada, and Mr. Ryoichi Nakajo from Ogata Lab. with whom I have conducted really exciting cognitive robotics studies in Waseda University. The everyday discussion on their research contents and chats with them have made my life as a doctoral student all the more enjoyable. I am thankful to the other Sugano Lab. and Ogata Lab. members for their comments,

their encouragement, and their interest in my research.

I would like to thank the members of Cognitive Neuro-Robotics Lab. in KAIST, especially, Mr. Minju Jung and Mr. Gibeom Park who are the first graduate students belonging to the newly established Prof. Tani's laboratory in KAIST. The life in a foreign country as a visiting student in this laboratory with them was very impressive and influential experiences.

I would like to thank the following secretaries in Waseda Univ.: Ms. Kyoko Arai and Ms. Yoko Ono of Sugano Lab., Ms. Naomi Nakata and Ms. Junko Inaniwa of Ogata Lab. I would also like to thank Ms. Junghyun Yoon, Ms. Jungmin Park, and Ms. Sarah (Seung-A) Oh who supported me not only to conduct research in KAIST but also to live in Korea as a foreigner.

I would like to acknowledge the *Yoshida Scholarship Foundation* for their financial support. My living expenses and graduate school fees for the three years as a doctoral student have been fully supported by their great scholarship, *Doctor 21*. A part of my international research activities including research stays in KAIST and participation in international conferences was also supported by this scholarship. I would also like to thank the *Hara Research Foundation* for their grant for participating in the 24th International Conference on Artificial Neural Networks (ICANN 2014).

Last but not least, I would like to express my gratitude to my friends and family for their ongoing support and encouragement, especially, to my parents who have allowed me to do whatever I want to do from childhood up to now.

Tokyo, January 18, 2016 Shingo Murata

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

One of the most fascinating human cognitive abilities is adaptive and flexible behavior generation together with adequate action, perception, and attention in dynamic and uncertain environment. Such an ability must also be necessary for artificial agents, for example, intelligent robots [1–3] that are expected to provide livelihood support in everyday life. Therefore, acquisition of the ability to generate adaptive and flexible behavior is a crucial issue for these robots situated in dynamic and uncertain real environment in contrast to industrial robots that only need to play-back pre-designed motions.

Looking at the developmental process of human infants as an example of intelligent agents, they gradually acquire cognitive abilities through dynamic interactions with their physical and social environment. In particular, at the early stage of the developmental process, interactions with their caregivers who actively support them are essential to achieve cognitive behavior such as playing with objects [4], imitation [5], and joint attention [6]. They further acquire more social cognitive skills through playing with other infants or children who may sometimes violate their expectations about others' behavior. What human infants learn through these interactions is what will happen when they do something, namely, the relationship between causal states and their consequences [7]. This can be cast as predictive learning of sensory consequences including exteroceptive (or visual) and proprioceptive states which are caused by the self and the external world including others [8–12]. The predictive learning is enabled by so-called *internal* or *generative models* of the world that are considered to be acquired in the brain [13–16]. A study on acquisition of generative models is crucial for

both understanding brain mechanisms of biological agents and implementing such mechanisms in artificial agents.

This thesis explores a generic computational framework for generative models which enables artificial agents or intelligent robots to achieve adaptive and flexible behavior.

1.1.1 Artificial Intelligence for Robotic Systems

The classical approach for realizing an internal model or representation for agents is to design and implement itself with symbolic representation by considering possible situations of the external world. This research paradigm based on the manipulation of symbols is referred to as *symbolic artificial intelligence (AI)* or *classical AI* and was dominant by the 1980s. One of the most crucial problems of the symbolic AI is the *frame problem* [17, 18]. This problem describes that a robot with a set of if-then rules designed by a human is necessary to renew the rules whenever the assumed surrounding environment changes even just a little. Another crucial problem is the *symbol grounding problem* [19] that describes the difficulty in the gap between discrete symbolic representations and their objectives in the continuous physical world with fluctuations.

Brooks [20] proposed an alternative approach, which is diametrically opposed to the symbolic AI, based on the so-called *subsumption architecture* [21] without any symbolic representation. This approach referred to as the *behavior-based approach*, also known as *nouvelle AI* or *embodied intelligence*, enhanced the importance of physical interaction between agents and the surrounding environment, and realized a certain level of intelligence without representation [22]. The behavior-based approach with the concept of embodiment had played a crucial role in the research field of intelligent or autonomous robots and originated a new research paradigm referred to as *embodied cognitive science* integrating cognitive science, psychology, artificial intelligence, and robotics [23]. Although robots developed based on this approach can produce behavioral patterns that are observed as intelligent behavior, the patterns are limited to reactive behavior due to the lack of higher-order cognitive functions such as planning and situational judgement.

1.1.2 Cognitive Robotics

Cognitive robotics, also referred to as *cognitive developmental robotics* [24–26]

and *cognitive neurorobotics* [27], has become popular since the 2000s. This is a new research paradigm to elucidate issues regarding the developmental process and underlying neural mechanism of human cognitive abilities in a synthetic or constructive manner. This research area, which also enhances the importance of embodiment, aims to provide computational models and mathematical hypotheses with reference to knowledge achieved by analytic approaches such as developmental psychology and cognitive neuroscience. The essential difference between behavior-based approach and cognitive robotics is that the latter tries to understand the developmental or learning process of higher-order cognitive functions by providing a minimal computational framework with a learning algorithm to artificial agents or humanoid robots that interact with the external world.

There are two research streams that are usually considered independently but closely related to cognitive robotics, namely, *theoretical neurobiology* [28, 29] and *robot learning* [30, 31]. Both the approaches emphasize the importance of machine learning perspective, however, their research focuses are totally different. Theoretical neurobiology, also known as computational neuroscience, focuses on theoretical understanding of brain or neural mechanisms of biological agents in both microscopic and macroscopic levels through numerical simulation. On the other hand, robot learning, also known as robot *programming by demonstration* (PbD) or *learning from demonstration* (LfD), focuses on developing learning algorithms for enabling artificial agents (robots) to achieve generalized skills based on acquired and embodied intelligence through their own experiences in real environment instead of pre-designed intelligence. In other words, the former emphasizes a scientific perspective, while the latter emphasizes an engineering or practical perspective.

Cognitive robotics trying to develop computational models that can be operated in real robotic systems may bridge the gap between these distinct scientific and engineering research streams. Especially, this thesis tries to apply the recent perspective from theoretical neurobiology through cognitive robotics to robot learning. Key studies in each research area are reviewed in a later section.

1.1.3 Action Primitives and Action Sequences

As an example, let us consider a situation where a boy is trying to carry a full glass of water from one place to another. He must pay attention to the glass

for delicately reaching for it with a precise hand positioning, and after grasping it he modulates his posture in order to avoid spilling the water. At the same time, he also needs to carefully plan a path toward a target place before actual movement if the floor is cluttered with his toys. He may start moving without any specific path planning and determine it in a reflexive manner as the occasion arises. Perhaps, he may drink a little on the way to avoid accidental spilling. Furthermore, if the environment includes others who may suddenly appear in an unpredictable manner, he needs to avoid collision with them by dynamically recognizing the environmental change. After arriving at the target place through a series of complicated situations that require dynamic cognitive processes of action, perception, and attention, he finally puts down the glass in a gentle manner on a target position. It should be noted that this situation can also be applied to an artificial agent (e.g. a humanoid robot) instead of a boy.

One of the important aspects considered in the above situation is that the whole action sequence can be understood as a combination of reusable *action primitives* such as "reaching for a target object," "grasping the object," "walking toward a target place," and "putting the object." These action primitives can be flexibly combined into an action sequence based on intentional states of agents or on sensory inputs derived from environmental changes. The idea to combine primitives to form a sequence was proposed by Arbib in terms of *schema theory* [32,33] in which primitives are called *schemas* and combinations or sequences of primitives are called *schema assemblages*. Another important aspect is that each action primitive should be acquired as a generalized skill that can be adaptively applied to novel situations. For example, the action primitive of reaching is adaptively modulated based on the position of a target object and flexibly followed by another primitive such as grasping the object or pushing it toward the opposite side.

1.1.4 Intra-Primitive and Inter-Primitive Variations

The adaptability and flexibility of action primitives, which enable humans to produce various types of complicated behavior, are acquired through experiences of interacting with the world including others. Repetitive experiences or trials in similar situations provide humans to extract task constraints that represent characteristics of action primitives and action sequences. The action primitive of reaching, for example, has a particular constraint or goal such that a hand finally

must be close to a target object. This invariant relationship between the final hand position and the target object position across trials may be the most crucial representation for specifying the skill of reaching. In other words, other features such as the path of the hand and the speed of the movement can vary depending on the situation in the reaching context. In what follows, these variations in each primitive are called *intra-primitive* variations, and the other variations that are produced by combining primitives are called *inter-primitive* variations.

The above account for specifying a skill in terms of the difference in the level of intra-primitive variations is similar to the concept of *global dynamics* for design of human and humanoid robot motion suggested by Kuniyoshi and Nagakubo [34]. The analysis of human rising motion [35, 36] revealed that there are invariant features or points called *nodes* and variant features or regions called *envelopes* in the phase space of body dynamics. Experiments with a simulated humanoid robot and an actual one also exhibited such nodes corresponding to critical conditions for the success of the rising motion [36]. These results imply that a precise controlling of the motion at each node and a relatively relaxed controlling in other regions may be crucial for achieving a task motion and for adapting to environmental conditions, respectively. A relevant psychological experiment [36] suggests the importance of the invariant features not only for action generation but also for action recognition. In the experiment, subjects were asked to watch a set of movie clips each of which showed a human rising motion with a different temporal length and to judge whether the performance was success or failure. The experimental results demonstrated that the inclusion of the invariant features to the clips enhances the performance of human action recognition.

1.1.5 Reactive Behavior and Proactive Behavior

Inter-primitive variations in which an adaptive action primitive is flexibly combined with another are realized by two distinct behavior generation schemes, namely, reactive behavior and proactive behavior[1]. These two behavior generation schemes are distinguished based on the origin of their causes [40] to de-

[1]There is another behavioral distinction, for example, between *habitual behavior* and *goal-directed behavior* [37]. The former behavior can be acquired by *model-free reinforcement learning* (RL) approaches and the latter by *model-based* RL approaches [37–39]. It should be noted that the behavioral distinction in this thesis is different from this distinction in terms of presence or absence of specific intentional states representing whole visuo-proprioceptive consequences of actions.

termine the next action primitive. The former generation scheme corresponds to exogenously formed behavior in which action primitives are determined by external causes such as sensory inputs of the moment [41, 42]. In contrast, the latter scheme corresponds to endogenously formed behavior in which whole action sequences are represented by particular intentional states[2] (internal causes) of agents [10, 44]. In the case of proactive behavior generation, different action sequences can be produced in terms of predictions (prior expectations) about visuo-proprioceptive consequences of actions depending on the intentional states. During reactive behavior generation, action generation will be delayed whereas during proactive behavior generation, an over dependence on own prediction can lead to inflexibility in action modification when the prediction fails.

In the context of robot behavior, reactive behavior has been considered in the behavior-based approach (introduced in Section 1.1.1) in which agents' actions are generated by sensory-motor mapping [45, 46] or sensory-motor coordination [47] without any contextual information. On the other hand, proactive behavior has been considered in model-based (learning) approach [48] in which agents' actions are determined by their context-dependent internal representations of the external world [49, 50]. Both the approaches have advantages and disadvantages. Behavior-based approach is suitable for producing simple robot behavior such as adaptive wall-following and collision-avoidance that can be determined by the current sensory state. However, this approach is unsuitable to deal with higher-order cognitive functions such as planning and situational judgement depending on the context which require agents to integrate past experiences, the current situation, and future states. On the other hand, model-based approach enables agents to realize these functions by acquiring the model through learning processes [51]. One crucial problem of this approach is its inflexibility when the model prediction fails and it does not fit to the current situation as mentioned above. This problem implies the necessity of dynamic model optimization or at least dynamic optimization of internal representations in the model-based approach.

Given these perspectives on both behavior-based approach for reactive behavior and model-based approach for proactive behavior, if we can handle these approaches in a unified framework, the usability of robotic systems may increase.

[2]More specifically, intentional states here mean *prior intention* introduced by Searle [43] to distinguish from the other intention called *intention in action*.

1.2 Research Questions and Hypotheses

Considering the adaptive and flexible behavior generation described in the preceding section, essential research questions that have not been revealed yet are twofold:

- How can artificial cognitive agents acquire adaptive action primitives together with the difference in the level of intra-primitive variations?
- How can these agents develop the distinct behavioral generation schemes for flexibly combining primitives into an action sequence together with inter-primitive variations?

This thesis tackles these two questions by focusing on the importance of predictive learning with *uncertainty estimation* together with the following two hypotheses:

- Estimation of *trajectory-level* uncertainty contributes to specifying action primitives that can be modulated based on environmental changes.
- Estimation of *event-level* uncertainty contributes to developing the distinct behavior generation schemes that produce various action sequences.

The remaining two subsections describe the reason for formulating these two hypotheses.

1.2.1 Importance of Uncertainty Estimation

At the beginning of this chapter, the importance of a generative model that can be acquired through the predictive learning of sensory consequences of action generation is pointed out. As noted above, this thesis hypothesizes that the generative model needs to learn to predict not only sensory consequences but also uncertainty of these consequences in both trajectory and event levels.

Regarding the acquisition of adaptive action primitives, estimation of trajectory-level uncertainty may be essential because intra-primitive variations can be described by the uncertainty about the time-varying visuo-proprioceptive trajectory representing a specific primitive. Invariant and variant parts in the trajectory are represented by low and high trajectory-level uncertainty, respectively.

On the other hand, regarding the development of the distinct behavioral generation schemes for flexibility, estimation of event-level uncertainty may be essential

because inter-primitive variations can be produced by transition of primitives (or events) from one to another and the transition can be described by the uncertainty about the next event such as transition probability. The next subsection further considers the aspect of the event-level uncertainty together with the relationship between differences in estimation of the uncertainty and the behavior generation schemes.

1.2.2 Event-Level Uncertainty and Behavior Generation

In human case, it can be considered that individuals construct their own interpretation of experiences or observed events through learning processes [7, 52]. In particular, when sensory events are observed as occurring probabilistically, there could be two interpretations by estimating event-level uncertainty in a different manner. One assumes a deterministic causal rule from the background or the context of the current sensation without considering any event-level uncertainty and the other assumes a probabilistic rule with considering a particular event-level uncertainty.

For example, let us suppose that one has already observed two sequences "AB" and "AD" where A, B and D are sensory events. When one next receives A, a probabilistic rule with event-level uncertainty would predict the occurrence of B or D with equal probability. On the other hand, if one uses a deterministic rule without any event-level uncertainty, then the prediction of both occurrences would be made deterministically by inferring a distinct background or context for each case. More specifically in the current example, if different contexts C′ or C″ can lead to the observation of A, the prediction of the next sensory state as B or D is made deterministically depending on the context inferred. The problem here is ill-posed because one can use both types of rule with different estimation of event-level uncertainty even though the past experience is exactly the same.

It is presumed that the choice to use the probabilistic rule or the deterministic rule would affect significantly the method of behavior generation by agents while interacting with the world and with others. If the next sensory state can be predicted only in a probabilistic manner, agents would generate reactive behavior in which the next action primitive to be taken will be determined optimally after the sensation is confirmed, as reaction to observed events. On the other hand, if the next and further sensation can be predicted deterministically with confidence, agents would generate proactive behavior in which the next and suc-

ceeding actions would be generated proactively based on a particular intentional state without waiting for the sensory input. This thesis considers how agents estimate event-level uncertainty in a different manner and how the differences influence the development of the behavior generation schemes in a self-organized fashion.

1.3 Research Objective

As we have seen in earlier sections, acquisition of generative models via predictive learning with uncertainty estimation may be crucial for achieving adaptive and flexible behavior. Generative models are known to be formulated in terms of a *predictive coding* [53] framework considering both action and perception, which is also called *predictive processing* [54]. Recurrent neural networks (RNNs) [55–58] have been adopted as one of possible computational frameworks to specify generative models thanks to their input- and context-dependent predictive learning capabilities under the simple computational principle of *prediction error minimization* [8, 9, 59]. RNNs can learn to generate predictions about the next state of target temporal sequence data by receiving the current state of those as input and incorporating contextual dynamics stored in the networks.

Conventional RNN-based frameworks, however, face three potential problems or issues due to their deterministic properties by which all data structures are modeled as deterministic dynamical systems even though the data are the successive states of stochastic processes. The first is that if RNNs are forced to learn multiple temporal sequence data with stochastic or random fluctuations, the learning process tends to become unstable with the accumulation of errors. The second is their inability to learn to extract and reproduce stochastic structures, such as the amplitude of the fluctuations, latent in the data. These two issues regarding the trajectory-level uncertainty are tightly coupled and important to consider the acquisition of adaptive action primitives as explained in Section 1.2.1. The third is that RNNs cannot estimate the event-level uncertainty that is important to consider the development of reactive and proactive behavior for realizing flexible behavior as explained in Section 1.2.2.

The objective of this thesis is to develop a novel RNN-based framework for generative models which can solve these three non-trivial issues and enables artificial cognitive agents to achieve adaptive and flexible behavior via predictive learning with uncertainty estimation. A series of studies in this thesis especially

tries to demonstrate the following two capabilities of the developed framework by conducting experiments on robot learning:

- Extraction of trajectory-level uncertainty via predictive learning and reproduction of learned skills (or action primitives) adaptively in unlearned situations
- Extraction of event-level uncertainty via predictive learning, reproduction of learned action sequences with the distinct behavior generation schemes, and flexible modification of the action sequences

The subsequent section reviews key concepts derived from studies in theoretical neurobiology, cognitive robotics, and robot learning which are tightly related to the present study.

1.4 Related Work

Predictive processing has gained widespread acceptance as one unified framework accounting for human cognitive aspects such as action, perception, attention, and learning. This framework has been developed in both cognitive neurorobotics with a connectionist scheme [8,9,59] and theoretical neurobiology with a Bayesian scheme [11,29] under the principle of prediction error or free energy minimization [10, 28, 60, 61]. The first subsection reviews these schemes.

The second subsection reviews studies about skill acquisition by artificial agents in both cognitive neurorobotics and robot learning. Especially, we focus on how uncertainty has been dealt with in these research areas.

1.4.1 Predictive Processing

Connectionist Scheme

Tani and colleagues [8, 9, 59, 62] proposed a deterministic connectionist scheme using an RNN-based hierarchical generative model, called *RNN with parametric biases* (RNNPB). RNNPB can learn to map top-down priors to predictions about visuo-proprioceptive consequences of an action by means of prediction error minimization. The parametric biases (PBs) are higher-level static vectors corresponding to the top-down priors that determine the characteristics of the forward dynamics of a lower-level network in a manner similar to the bifurcation parameters, also known as control parameters, of nonlinear dynamical systems.

They demonstrated that learning, generation, and recognition of multiple visuo-proprioceptive consequences of actions produced by a robot can be formulated as prediction error minimization by using RNNPB. Under this formulation, the learning of action sequences is the process of optimizing network parameters including synaptic weights, biases that are shared by all sequences, and the PBs that are specific to each sequence, in order to regenerate the action sequences given visuo-proprioceptive sequences. After the learning process, a robot equipped with the trained network can regenerate each learned action sequence in a top-down manner based on the corresponding PB value, which represents a top-down prior, by sending the predicted proprioceptive state to the motor controller as the next target state of the robot.

During top-down behavior generation, a dynamic recognition process can also be performed in a bottom-up manner by inferring the PB value that can regenerate a given part of visuo-proprioceptive states through prediction error minimization with fixed weights and biases. More specifically, generated prediction errors in a "window" spanning the immediately preceding steps are back-propagated to the PB units and the current PB states are modulated in the direction of minimizing the prediction errors. This scheme for the dynamic recognition process is referred to as the *error regression scheme* (ERS). The ERS introduces a fundamentally different sort of inference scheme in which the prediction errors are dynamically minimized online by modulating the internal neural dynamics with not forward but backward dynamics.

Bayesian Scheme

Friston and colleagues [10,11,29] proposed a Bayesian scheme, called *active inference*, which entails the Bayesian brain hypothesis [13, 63] and is based on a *free-energy principle* [28,64,65]. Active inference can be organized by a hierarchically structured probabilistic generative model in which neural states at higher levels provide empirical priors on lower-level states in a top-down manner, and lower-level states provide prediction errors to higher levels for inference in a bottom-up manner. Under this scheme, prediction errors can be reduced by changing the externally given sensory signals being predicted and the internally generated predictions, through action and perception, respectively. As described in [11, 27], active inference is a generic Bayesian perspective on the above mentioned connectionist scheme using RNNPB. The key aspect of this Bayesian approach is

the ability to deal with uncertainty or precision, which has been related to attentional mechanisms [28, 66–69]. The implicit estimation of uncertainty has not been considered in the deterministic connectionist scheme.

1.4.2 Uncertainty in Robot Learning

Learning with Deterministic Frameworks

Tani and colleagues [44,70,71] demonstrated that not only deterministic sequences but also probabilistic transition sequences can be embedded in RNN-based deterministic models. In the context of action imitation learning by cognitive agents, Namikawa et al. [44] showed that a functional hierarchy [72, 73], which accounts for spontaneous behavior generation, can be self-organized in a multiple timescale RNN (MTRNN) [74]. The MTRNN consisted of a lower-level network containing a set of action primitives with fast dynamics and a higher level with slow dynamics that drove the lower-level network for combining the primitives. In their experiments, a humanoid robot controlled by the trained network was able to generate "pseudo-stochastic" action sequences by deterministic chaos self-organized in the higher-level network. When the transition probabilities of training data or observed events were changed, the network was able to reconstruct the probabilities in a deterministic manner by using the self-organized chaotic dynamics. Although this can be considered as one approach to the generation of stochastic sequences, it only models proactive behavior generation and does not consider reactive behavior. Furthermore, the problem of inflexibility in action modification has not been addressed.

Learning with Probabilistic Frameworks

In the context of PbD or LbD, Calinon and Billard greatly contributed by using probabilistic frameworks such as *hidden Markov model* (HMM) with interpolation [75–78] and the joint use of *Gaussian mixture model* (GMM) and *Gaussian mixture regression* (GMR) [30]. One of important aspects of their probabilistic frameworks using GMM is the ability to encode the uncertainty of multiple demonstration of behavioral trajectories in terms of covariance matrix. In their approach, GMM is used for encoding a set of behavior trajectories with mean and covariance matrix and GMR is used for retrieving a smooth generalized version of these trajectories and associated variables. Although continuous task constraints

can be represented by the time-varying covariance matrix, the crucial problem is the sensitivity to temporal and spatial perturbations due to the time-dependency.

As alternative approaches, recently dynamical systems approaches such as *stable estimator of dynamical systems* (SEDS) [79–81] and statistical dynamical systems [82] have been adopted. These dynamical systems approaches are good at encoding primitive actions, however, methods to flexibly combine these primitives have not been established.

1.5 Overview of Proposed Approach

This thesis proposes a novel computational framework that enables RNNs to learn to predict both the mean and the variance of the next state of target data with fluctuations, where the variance corresponds to the uncertainty of target variables and the reciprocal of the variance is called precision. Studies described in this thesis especially consider the implementation of this framework on conventional continuous-time RNNs (CTRNNs) whose context states employ leaky-integrator neural units to deal with continuous data flow. The novel network is referred to as stochastic CTRNN (S-CTRNN) in terms that an aspect of stochasticity is introduced into the conventional CTRNN.

The additional variance prediction mechanism enables the S-CTRNN (1) to stably learn multiple temporal sequence data with random fluctuations because the learning process tries to minimize prediction errors scaled by the predicted variance, which are called precision-weighted prediction errors, and (2) to extract and reproduce stochastic structures latent in such data in terms of the time-varying mean and variance states. These two points contributes to solve the first and second issues described in Section 1.3. For the third issue, an extension of the S-CTRNN, which is inspired by the MTRNN introduced in Section 1.4.2, is considered. The extended hierarchical network referred to as stochastic multiple timescale RNN (S-MTRNN) consists of the lower-level and higher-level subnetworks with different timescale dynamics of the context states. The S-MTRNN can self-organize internal representations of primitives in the lower-level subnetwork with fast dynamics and those of sequences of the primitives in the higher-level subnetwork with slow dynamics. By incorporating this ability to self-organize hierarchical representations and the variance prediction mechanism, the S-MTRNN is able (3) to determine its modeling way, namely, probabilistic or deterministic modeling, depending on the condition of learning. A novel er-

ror regression scheme (ERS) for the dynamic recognition considering uncertainty by the S-MTRNN, which is inspired by the ERS for the RNNPB introduced in Section 1.4.1, is also proposed.

The abilities of these stochastic RNNs are verified through a series of numerical experiments and two kinds of robot experiments. The numerical experiments in which artificial training data are generated in different ways utilizing Gaussian noise consider the first and second issues regarding the trajectory-level uncertainty. The S-CTRNN demonstrates its ability to stably learn to extract and reproduce stochastic structures latent in these data. The first robot experiment on learning reaching movement toward a fixed target object from human demonstrations via kinesthetic teaching also considers these issues in more realistic situations. The experimental results demonstrate the development of adaptive behavior by estimating the trajectory-level uncertainty. The second robot experiment on learning interactive behavior with another robot considers the third issue regarding the event-level uncertainty. The experimental results demonstrate the development of flexible behavior based on reactive and proactive behavior by estimating the event-level uncertainty.

1.6 Organization of the Thesis

Figure 1.1 shows the organization of the thesis. Chapter 2 proposes a new computational framework that enables RNNs to deal with stochasticity. Chapter 3 demonstrates the results of a series of numerical experiments on learning and reproduction of temporal sequence data with random fluctuations. Chapter 4 demonstrates the results of a robot experiment on learning to develop adaptive behavior from human demonstrations via kinesthetic teaching. Chapter 5 demonstrates the results of a robot experiment on learning to develop flexible behavior via interaction with another robot. Chapter 6 summarizes the achievements of a series of both numerical and robot experiments. The reviews on remaining issues and on future directions conclude this thesis.

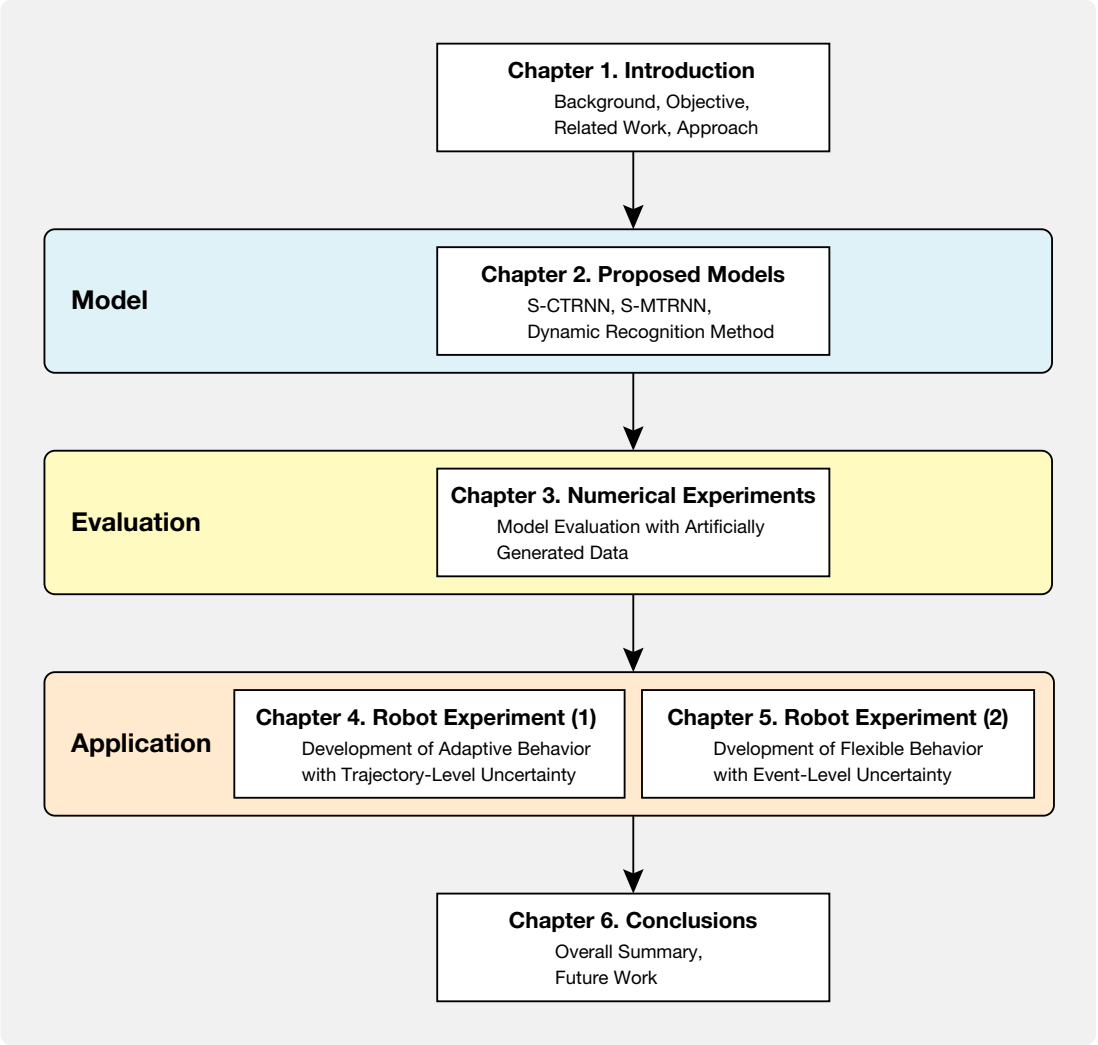


Figure 1.1 Organization of the thesis.

Chapter 2

Stochastic Recurrent Neural Networks

2.1 Introduction

This chapter proposes a novel computational framework that enables the conventional CTRNNs to learn to predict not only deterministic but also stochastic (or fluctuating) temporal sequence data. The proposed framework is based on the predictive learning with uncertainty estimation and the CTRNN with this framework is referred to as stochastic CTRNN (S-CTRNN) from the ability to deal with stochasticity of target data. The predictive learning with uncertainty estimation contributes to the following three issues: (1) the stability in the learning of temporal sequence data with random fluctuations, (2) the reproduction of the stochastic structures latent in the learned data, and (3) the development of both reactive and proactive behavior in the context of robot learning.

This chapter also proposes an extension of the S-CTRNN, which is inspired by the MTRNN introduced in the preceding chapter (Section 1.4.2). The extended model with multiple timescale dynamics of the context units is referred to as stochastic MTRNN (S-MTRNN). In addition to the implementation of multiple timescale dynamics, this chapter considers a novel error regression scheme (ERS) together with the uncertainty estimation for the dynamic recognition by the S-MTRNN, which is inspired by the ERS for the RNNPB introduced in the preceding chapter (Section. 1.4.1).

Numerical and robot experiments with the S-CTRNN mainly regarding the first and second issues are presented in Chapters 3 and 4, and a robot experiment with the S-MTRNN mainly regarding the third issue is presented in Chapter 5.

The next section reviews conventional artificial neural networks including feedforward neural networks (FNNs) and RNNs as a preliminary step toward the proposed framework detailed in the sections after the next.

2.2 Review of Conventional Neural Networks

This section provides a review of conventional artificial neural networks including FNNs and RNNs. Their computational frameworks are detailed in Appendix A.

2.2.1 Feedforward Neural Networks (FNNs)

FNNs, also known as multi-layer perceptrons (MLPs), can learn non-linear input–output function by using training data consisting of multiple sets of input–target observations. These networks can be applied for both classification problems in which target states are discrete categories and regression problems in which target states are continuous values. Computational models for classification problems are called *discriminative model* and for regression problems *generative model*.

FNNs consist of three kinds of layers as shown in Fig. 2.1A: an input, a hidden, and an output layer, each of which includes some neural units. More detailed illustration representing a network diagram of an FNN is provided in Fig. A.1 in Appendix A (Section A.1). The units in the input layer have connections from them to the units in the hidden layer. The units in the hidden layer have connections from them to the units in the output layer. Gradient-based predictive learning of FNNs can be conducted by using the back-propagation (BP) algorithm [83] as detailed in Appendix A (Section A.1.2).

2.2.2 Recurrent Neural Networks (RNNs)

The FNNs introduced in the preceding subsection have a potential problem of their inability to deal with context dependency. In order to overcome this problem, RNNs [55–58] were developed. The essential difference between FNNs and RNNs is their neural units in the hidden layer. The units of the hidden layer in RNNs have recurrent connections by which networks can memorize past states. In terms of the fact that the hidden layer can deal with contextual information, the layer is called *context layer* instead of hidden layer. It should be noted that RNN whose context states employ leaky-integrator neural units to deal with continuous data flow is called continuous-time RNNs (CTRNNs). Architecture of an

RNN is illustrated in Fig. 2.1B. More detailed illustration representing a network diagram of an RNN is provided in Fig. A.2 in Appendix A (Section A.2).

Temporal processing of the FNN and that of the RNN is provided in Fig. 2.1C and D, respectively, where the recurrent connection of the RNN is unfolded through time. From this illustration, we can understand that the input information at the time step $t = 1$ is stored in the context state at the final time step $t = T$ thanks to the recurrent connections. Gradient-based predictive learning of RNNs can be conducted by applyig the BP algorithm to the unfolded RNN, which is called back-propagation through time (BPTT), as detailed in Appendix A (Section A.2.2).

2.2.3 Sensitivity to Initial Conditions

When learning temporal sequences with RNNs through optimizing the parameters consisting of synaptic weights and biases, context sensitivity can be modeled using sensitivity to initial conditions of the internal neural dynamics. By optimizing specific initial internal states of the context units (the leftmost state in Fig. 2.1D) for each sequence, multiple attractors such as fixed point, limit cycle, and strange (chaotic) attractors can be self-organized in a single CTRNN [84]. Heuristically, the initial conditions encode different contexts by starting in different basins of attraction that give rise to attractor dynamics with distinct forms. In deterministic chaos, it is well known that small differences in initial conditions can yield widely diverging state trajectories.

In the context of robot learning, Nishimoto et al. [85] showed that three essential functions of learning, generation, and recognition can be achieved by using the sensitivity to initial conditions or initial precision characteristic of the context states of a CTRNN. This characteristic is also related to biological brains. For example, a monkey electrophysiological study [86] suggests that preparatory activity in motor and premotor cortex sets the initial states of a neural dynamical system whose evolution produces reaching movement activity.

2.3 Overview of Stochastic Continuous Time RNN (S-CTRNN)

In the preceding section, the conventional artificial neural networks including FNNs and RNNs (CTRNNs) were reviewed. Especially, CTRNNs are well-known

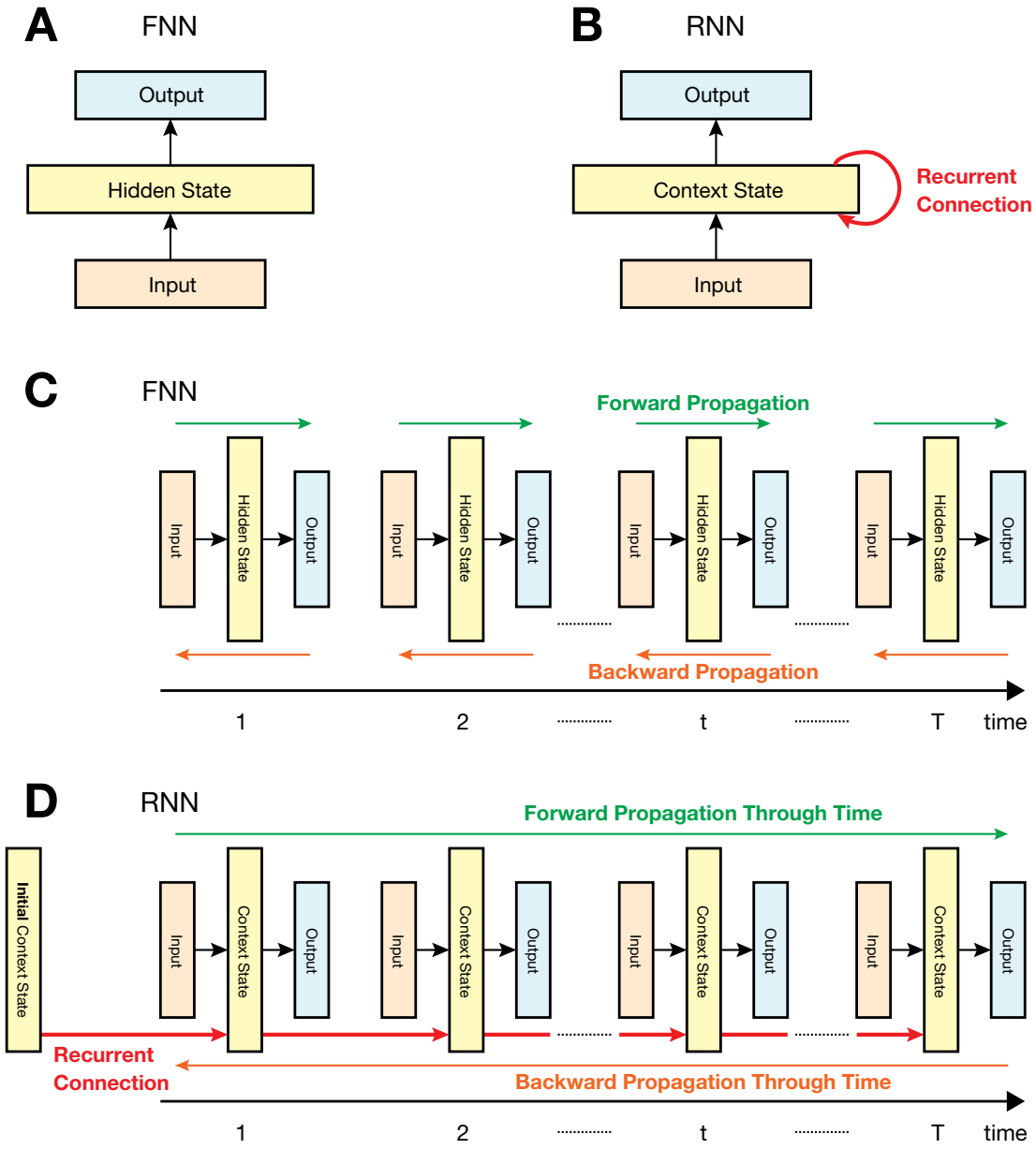


Figure 2.1 Comparison of architectures and temporal processing between FNN and RNN. Architecture of (A) an FNN and (B) an RNN. The FNN consists of an input layer, a hidden layer, and an output layer. The RNN consists of an input layer, a context layer with recurrent connections, and an output layer. The architecture of the RNN is the same as that of the FNN except for the recurrent connection in the context layer. Temporal processing of (C) the FNN and (D) the RNN. Information flow of forward propagation is colored with green and that of backward propagation is colored with orange. The recurrent connection of the RNN by which the network can deal with context dependency are unfolded through time. Context states at each time step receive weighted sum of the input states at that time step and context states at the previous time step. Especially, the context states at the time step $t = 1$ receive weighted sum of the initial context states that can be optimized in the learning process. These initial context states have a role to determine the forward dynamics by utilizing the characteristic of non-linear dynamical systems referred to as *sensitivity to initial conditions* or *initial precision characteristic*. By utilizing this characteristic, multiple attractors such as fixed point, limit cycle, and chaotic attractors can be embedded into a single RNN.

to be powerful tools for learning to predict various types of temporal sequence data [46,87,88]. Due to the deterministic computational frameworks based on the prediction error minimization, however, these networks have potential problems of their inability to deal with target data with stochastic or random fluctuations which cannot be predicted in a deterministic manner.

The S-CTRNN makes use of a novel feature manifested by additional neural units allocated in the *variance layer* whose states are mapped from the context layer as well as the output states. By utilizing these states, the network predicts not only the mean of the next input states in the output layer, but also their variance states in the variance layer. In this method, the mean and the variance can be obtained by means of maximization of the likelihood function for network parameters. Furthermore, upon achieving convergence of the likelihood, the network can reproduce temporal sequence data with the same stochastic structures as the fluctuating target sequence by adding Gaussian noise with the variance predicted at each time step to the predicted mean and subsequently feeding these values as input states. Figure 2.2 presents a comparison between the proposed S-CTRNN and the conventional CTRNN.

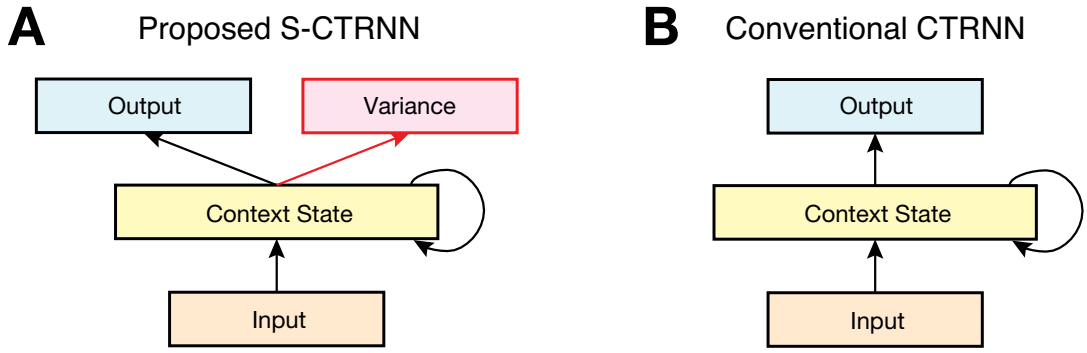


Figure 2.2 Comparison of architectures between (A) the proposed S-CTRNN and (B) the conventional CTRNN (already shown in Fig. 2.1B). Context states of the S-CTRNN are mapped to both the output and variance states. The additional parts compared to the conventional CTRNN are colored with red (the variance layer and the connection from the context layer). The other parts are the same as those for the CTRNN.

2.4 Form of Generative Model

When a set of fluctuating temporal sequence data is given, under Gaussian assumptions the ith dimension of the target state at time step t of the sth sequence

$(\hat{y}_{t,i}^{(s)})$ can be modeled in the form

$$\hat{y}_{t,i}^{(s)} \sim \mathcal{N}(y_{t,i}^{(s)}, v_{t,i}^{(s)}), \tag{2.1}$$

where $\mathcal{N}(y_{t,i}^{(s)}, v_{t,i}^{(s)})$ is Gaussian distribution with mean $y_{t,i}^{(s)}$ and variance $v_{t,i}^{(s)}$. The S-CTRNN as a generative model is trained to generate both the mean states $y_{t,i}^{(s)} = f(u_{t,i}^{(s)})$ as output states and the variance states $v_{t,i}^{(s)} = g(u_{t,i}^{(s)})$ by receiving current input states and using contextual dynamics stored in the network, where

$u_{t,i}^{(s)}$: internal state of the ith output or variance unit at time step t corresponding to the sth sequence,

$f(\cdot), g(\cdot)$: activation functions for the output and variance states, respectively.

Here, the variance corresponds to the uncertainty of taget states and the reciprocal of the variance is called precision.

2.5 Forward Propagation

Consider an S-CTRNN consisting of an input layer with N_{I}-dimensional units, a context layer with N_{C}-dimensional units, an output layer with N_{O}-dimensional units, and a variance layer with N_{O}-dimensional units as shown in Fig. 2.3. The forward dynamics of the internal states of the ith context, output, and variance unit at time step $1 \leq t$ corresponding to the sth target sequence $(u_{t,i}^{(s)})$ are given by

$$u_{t,i}^{(s)} = \begin{cases} \left(1 - \dfrac{1}{\tau_i}\right) u_{t-1,i}^{(s)} + \dfrac{1}{\tau_i} \left\{ \left(\displaystyle\sum_{j \in I_{\mathrm{I}}} w_{ij} x_{t,j}^{(s)} \right) + \left(\sum_{j \in I_{\mathrm{C}}} w_{ij} c_{t-1,j}^{(s)} \right) + b_i \right\} & (i \in I_{\mathrm{C}}), \\ \left(\displaystyle\sum_{j \in I_{\mathrm{C}}} w_{ij} c_{t,j}^{(s)} \right) + b_i & (i \in I_{\mathrm{O}} \cup I_{\mathrm{V}}). \end{cases} \tag{2.2}$$

where

τ_i: time constant of the ith context unit,

$I_{\mathrm{I}}, I_{\mathrm{C}}, I_{\mathrm{O}}, I_{\mathrm{V}}$: index sets for the input, context, output, and variance units, respectively,

$w_{i,j}$: synaptic weight of the connection from the jth unit to the ith unit,

$x_{t,j}^{(s)}$: the jth external input state at time step t corresponding to the sth targe sequence,

$c_{t,j}^{(s)}$: neural activation state of the jth context unit at time step t corresponding to the sth target sequence,

b_i: bias of the ith unit.

The neural activation states of context units $c_{t,i}^{(s)}$, output units $y_{t,i}^{(s)}$, and varaince units $v_{t,i}^{(s)}$ at time step t corresponding to the sth target sequence are calculated by using the respective activation function as follows:

$$c_{t,i}^{(s)} = \tanh(u_{t,i}^{(s)}) \qquad (0 \leq t \wedge i \in I_{\mathrm{C}}), \tag{2.3}$$

$$y_{t,i}^{(s)} = \tanh(u_{t,i}^{(s)}) \qquad (1 \leq t \wedge i \in I_{\mathrm{O}}), \tag{2.4}$$

$$v_{t,i}^{(s)} = \exp(u_{t,i}^{(s)}) \qquad (1 \leq t \wedge i \in I_{\mathrm{V}}). \tag{2.5}$$

2.6 Predictive Learning

The predictive learning of S-CTRNNs is conducted based on the maximum likelihood estimation by utilizing the gradient ascent method [89]. The learning process consists of the following phases.

1. Initialization of learnable parameters (synaptic weight w_{ij}, bias b_i, and initial internal states of context units $u_{0,i}^{(s)}$) (described in Section 2.7).
2. Generation of mean states $y_{t,i}^{(s)}$ and variance states $v_{t,i}^{(s)}$ by forward dynamics under the current parameter settings (described in Section 2.5).
3. Calculation of likelihood by using target states $\hat{y}_{t,i}^{(s)}$, and the mean and variance states predicted by the network.
4. Updating the parameters with the gradient ascent method.
5. Repeating phases 2-4 until the likelihood converges.

The current section introduces phases 3 and 4.

2.6.1 Objective Function

Here, the learnable parameters of the network (synaptic weight w_{ij}, bias b_i, and initial internal states of context units $u_{0,i}^{(s)}$) are denoted as $\boldsymbol{\theta}$. Suppose that we are given S target sequences, each of which consists of

$$\{\hat{\boldsymbol{y}}_t^{(s)}\}_{t=1}^{T^{(s)}} = \{\hat{\boldsymbol{y}}_1^{(s)}, \hat{\boldsymbol{y}}_2^{(s)}, \cdots, \hat{\boldsymbol{y}}_t^{(s)}, \cdots, \hat{\boldsymbol{y}}_{T^{(s)}}^{(s)}\}, \tag{2.6}$$

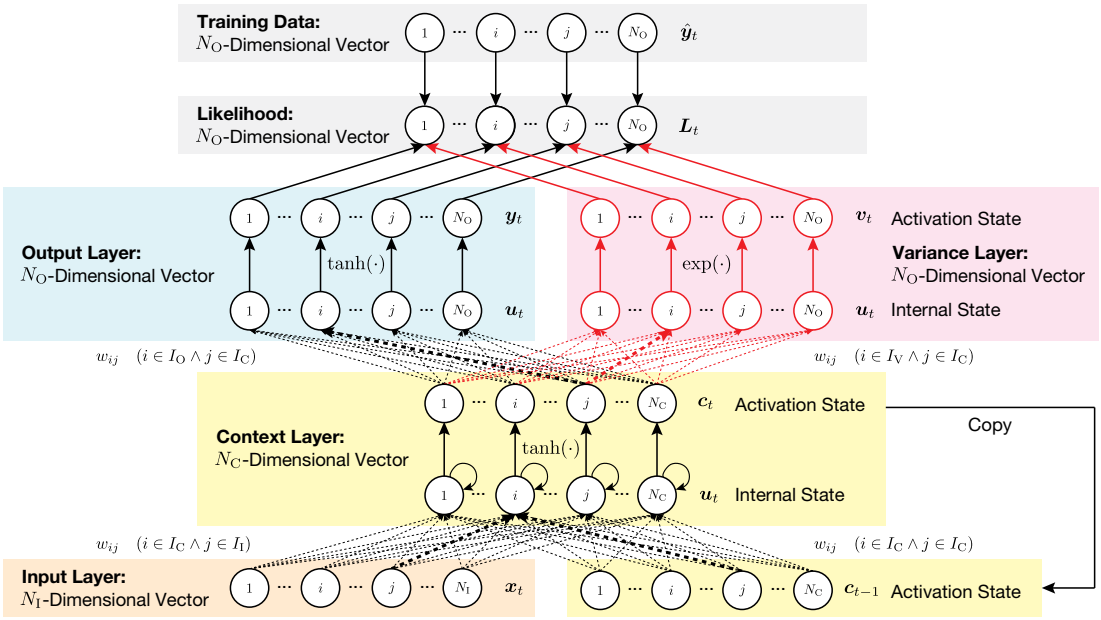


Figure 2.3 Network diagram of S-CTRNN. The input layer including N_{I}-dimensional neural units (nodes in the input layer) receives external input states $\boldsymbol{x}_t$. These units are connected with N_{C}-dimensional neural units (nodes in the bottom of the context layer) and internal states $\boldsymbol{u}_t$ of the context layer are computed as an weighted sum of the current input states and previous context activation states $\boldsymbol{c}_{t-1}$. Synaptic weights are represented by links (black dashed lines) between the nodes. The bold black dashed line indicates the connection from the jth to the ith unit (w_{ij}). The computed internal states are transformed to activation states $\boldsymbol{c}_t$ (nodes in the top of the context layer) by using the hyperbolic tangent function $\tanh(\cdot)$. In a similar way, internal states $\boldsymbol{u}_t$ of the output layer are computed as an weighted sum of context activation states and output activation states $\boldsymbol{y}_t$ are achieved by applying $\tanh(\cdot)$ to the internal states. In addition to the states of the output layer, internal states $\boldsymbol{u}_t$ of the variance layer are also computed as an weighted sum of context activation states and variance activation states $\boldsymbol{v}_t$ are achieved by applying $\exp(\cdot)$ to the internal states. By computing the product of the probability of training data under Gaussain assumption, we obtain the likelihood function of the network parameters ($\boldsymbol{L}_t$) that is defined by target states $\hat{\boldsymbol{y}}_t$, output activation states $\boldsymbol{y}_t$, and variance activation states $\boldsymbol{v}_t$. This likelihood function is used for the gradient-based predictive learning with BPTT. The additional parts compared to the conventional CTRNN shown in Fig. A.3 are colored with red (internal and activation states in the variance layer and connections related to the states). The other parts are the same as those for the CTRNN.

where $\hat{\boldsymbol{y}}_t^{(s)}$ is a N_{O}-dimensional target state vector whose ith element is represented as $\hat{y}_{t,i}^{(s)}$, s is the index of the sequence, and $T^{(s)}$ is the length of the sth sequence. The input to an S-CTRNN is desribed as $\boldsymbol{x}_t^{(s)} = \hat{\boldsymbol{y}}_{t-\xi}^{(s)}$ and an input sequence is given by

$$\{\boldsymbol{x}_t^{(s)}\}_{t=1}^{T^{(s)}} = \{\boldsymbol{x}_1^{(s)}, \boldsymbol{x}_2^{(s)}, \cdots, \boldsymbol{x}_t^{(s)}, \cdots, \boldsymbol{x}_{T^{(s)}}^{(s)}\}, \tag{2.7}$$

where $1 \leq \xi$ is called *delay length*. In this case, the probability density function of the target state $\hat{y}_{t,i}^{(s)}$ is defined as

$$p(\hat{y}_{t,i}^{(s)} \mid \{\boldsymbol{x}_{t'}^{(s)}\}_{t'=1}^{t}, \boldsymbol{\theta}) = \mathcal{N}(\hat{y}_{t,i}^{(s)} \mid y_{t,i}^{(s)}, v_{t,i}^{(s)}) \tag{2.8}$$

$$= \frac{1}{\sqrt{2\pi v_{t,i}^{(s)}}} \exp\left(-\frac{(\hat{y}_{t,i}^{(s)} - y_{t,i}^{(s)})^2}{2v_{t,i}^{(s)}}\right), \tag{2.9}$$

where $y_{t,i}^{(s)}$ and $v_{t,i}^{(s)}$ are the output and variance states generated by the network. This equation is derived from the Gaussian assumption explained in Section 2.4.

The likelihood function L_{out} parameterized by $\boldsymbol{\theta}$ is denoted by the following product of (2.8) with respect to the sequence s, the time step t, and the output dimension i:

$$L_{\mathrm{out}} = \prod_{s \in I_{\mathrm{S}}} \prod_{t=1}^{T^{(s)}} \prod_{i \in I_{\mathrm{O}}} p(\hat{y}_{t,i}^{(s)} \mid \{\boldsymbol{x}_{t'}^{(s)}\}_{t'=1}^{t}, \boldsymbol{\theta}). \tag{2.10}$$

The parameters $\boldsymbol{\theta}$ are optimized through the learning process in the direction to maximize the likelihood L_{out}. More precisely, we use the gradient ascent method with a momentum term as the procedure for the parameter otpimization. Here, the logarithm of the expression in (2.10) is used to facilitate the calculation.

$$\ln L_{\mathrm{out}} = \sum_{i \in I_{\mathrm{S}}} \sum_{t=1}^{T^{(s)}} \sum_{i \in I_{\mathrm{O}}} \left(-\frac{\ln(2\pi v_{t,i}^{(s)})}{2} - \frac{(\hat{y}_{t,i}^{(s)} - y_{t,i}^{(s)})^2}{2v_{t,i}^{(s)}}\right). \tag{2.11}$$

It is well known that minimizing the squared error is equivalent to maximizing the log-likelihood determined from Gaussian distribution. Up to constants, from (2.11) the negative log-likelihood $\mathcal{L}_{\mathrm{out}} = -\ln L_{\mathrm{out}}$ is given by

$$\mathcal{L}_{\mathrm{out}} = \sum_{i \in I_{\mathrm{S}}} \sum_{t=1}^{T^{(s)}} \sum_{i \in I_{\mathrm{O}}} \left(\underbrace{\frac{\ln v_{t,i}^{(s)}}{2}}_{\text{log-uncertainty}} + \underbrace{\frac{(\hat{y}_{t,i}^{(s)} - y_{t,i}^{(s)})^2}{2v_{t,i}^{(s)}}}_{\text{precision-weighted prediction error}} \right). \tag{2.12}$$

This equation means that the objective function for the predictive learning of S-CTRNNs consists of the sum of log-uncertainty and prediction error divided by the predicted variance or *precision-weighted prediction error*. By comparing the objective function for CTRNNs (A.22) and that for S-CTRNNs (2.12), we can understand the following key features: The S-CTRNN can avoid unstable learning of temporal sequence data with random fluctuations since the predicted variance $v_{t,i}^{(s)}$ functions as an inverse weighting factor for the prediction errorr (squared error) $(\hat{y}_{t,i}^{(s)} - y_{t,i}^{(s)})^2/2$. More specifically, the effect of the prediction error is reduced when the predicted 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 temporal sequences that display considerable fluctuations. This relaxes the predictive learning of stochastic temporal sequence data.

When a set of several sequences is used as training data, distinct initial internal states $\boldsymbol{u}_0^{(s)}$ must be provided for each sequence. We consider that the distribution of the initial internal states also conforms to Gaussian distribution. The probability density function for $u_{0,i}^{(s)}$, which is the initial internal state of the ith unit corresponding to the sth target sequence, is defined as

$$p(u_{0,i}^{(s)} \mid u_i, \sigma_{\mathrm{IS}}^2) = \mathcal{N}(u_{0,i}^{(s)} \mid u_i, \sigma_{\mathrm{IS}}^2) \tag{2.13}$$

$$= \frac{1}{\sqrt{2\pi}\sigma_{\mathrm{IS}}} \exp\left(-\frac{(u_{0,i}^{(s)} - u_i)^2}{2\sigma_{\mathrm{IS}}^2}\right), \tag{2.14}$$

where u_i is the mean value of the initial internal states, which is a learnable parameter, and σ_{IS}^2 is the predefined variance.

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

$$L_{\mathrm{init}} = \prod_{s \in I_{\mathrm{S}}} \prod_{i \in I_{\mathrm{C}}} p(u_{0,i}^{(s)} \mid u_i, \sigma_{\mathrm{IS}}^2). \tag{2.15}$$

The logarithm of the expression in (2.15) is used to facilitate the calculation.

$$\ln L_{\mathrm{init}} = \sum_{s \in I_{\mathrm{S}}} \sum_{i \in I_{\mathrm{C}}} \left(-\frac{\ln(2\pi\sigma_{\mathrm{IS}}^2)}{2} - \frac{(u_{0,i}^{(s)} - u_i)^2}{2\sigma_{\mathrm{IS}}^2}\right). \tag{2.16}$$

2.6.2 Gradient Ascent Method

The network parameters $\boldsymbol{\theta}$, consisting of synaptic weights, biases, initial internal states of the context units, and the mean values of these initial states, are

optimized to maximize the corresponding log-likelihood $\ln L$. The parameters at learning step n ($\boldsymbol{\theta}_n$) are updated by the gradient ascent method with a momentum term:

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \alpha\Delta\boldsymbol{\theta}_n, \tag{2.17}$$

$$\Delta\boldsymbol{\theta}_n = \frac{\partial \ln L}{\partial \boldsymbol{\theta}} + \eta\Delta\boldsymbol{\theta}_{n-1}, \tag{2.18}$$

$$\Delta\boldsymbol{\theta}_n = 0, \tag{2.19}$$

where α is the learning rate and η is a parameter representing the momentum term.

For updating the parameters $\boldsymbol{\theta}_{\text{share}}$, consisting of weights w_{ij} and biases b_i that are shared for the generation of all sequences, the likelihood L and the learning rate α in (2.19) are replaced with L_{out} and α_{share}, respectively. For updating the other parameters $\boldsymbol{\theta}_{\text{init}}$, consisting of the initial internal states of the context units $u_{0,i}^{(s)}$ that are provided for the generation of each sequence s and their mean values u_i, the likelihood L and the learning rate α in (2.19) are replaced with $L_{\text{all}} = L_{\text{out}}L_{\text{init}}$ and α_{init}, respectively. Details about the calculation of gradients $\frac{\partial \ln L_{\text{out}}}{\partial \boldsymbol{\theta}_{\text{share}}}$ and $\frac{\partial \ln L_{\text{all}}}{\partial \boldsymbol{\theta}_{\text{init}}}$ are provided in the subsequent section.

2.6.3 Back-Propagation Through Time

The gradient $\frac{\partial \ln L_{\text{out}}}{\partial \boldsymbol{\theta}_{\text{share}}}$ and $\frac{\partial \ln L_{\text{all}}}{\partial \boldsymbol{\theta}_{\text{init}}}$ for each learnable parameter can be obtained by applying the conventional BPTT method [83]:

$$\frac{\partial \ln L_{\text{out}}}{\partial w_{ij}} = \begin{cases} \dfrac{1}{\tau_i}\displaystyle\sum_{s\in I_{\text{S}}}\sum_{t=1}^{T^{(s)}} x_{t,j}^{(s)}\frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}^{(s)}} & (i \in I_{\text{C}} \wedge j \in I_{\text{I}}), \\ \dfrac{1}{\tau_i}\displaystyle\sum_{s\in I_{\text{S}}}\sum_{t=1}^{T^{(s)}} c_{t-1,j}^{(s)}\frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}^{(s)}} & (i \in I_{\text{C}} \wedge j \in I_{\text{C}}), \\ \displaystyle\sum_{s\in I_{\text{S}}}\sum_{t=1}^{T^{(s)}} c_{t,j}^{(s)}\frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}^{(s)}} & (i \in I_{\text{O}} \cup I_{\text{V}} \wedge j \in I_{\text{C}}), \end{cases} \tag{2.20}$$

$$\frac{\partial \ln L_{\mathrm{out}}}{\partial b_i} = \begin{cases} \dfrac{1}{\tau_i} \displaystyle\sum_{s \in I_{\mathrm{S}}} \sum_{t=1}^{T^{(s)}} \frac{\partial \ln L_{\mathrm{out}}}{\partial u_{t,i}^{(s)}} & (i \in I_{\mathrm{C}}), \\ \displaystyle\sum_{s \in I_{\mathrm{S}}} \sum_{t=1}^{T^{(s)}} \frac{\partial \ln L_{\mathrm{out}}}{\partial u_{t,i}^{(s)}} & (i \in I_{\mathrm{O}} \cup I_{\mathrm{V}}), \end{cases} \tag{2.21}$$

$$\frac{\partial L_{\mathrm{out}}}{\partial u_{t,i}^{(s)}} = \begin{cases} \left\{1 - (c_{t,i}^{(s)})^2\right\} \left(\displaystyle\sum_{k \in I_{\mathrm{C}}} \frac{w_{ki}}{\tau_k} \frac{\partial L_{\mathrm{out}}}{\partial u_{t+1,k}^{(s)}} + \sum_{k \in I_{\mathrm{O}} \cup I_{\mathrm{V}}} w_{ki} \frac{\partial L_{\mathrm{out}}}{\partial u_{t,k}^{(s)}} \right) & \\ \qquad + \left(1 - \dfrac{1}{\tau_i}\right) \dfrac{\partial L_{\mathrm{out}}}{\partial u_{t+1,i}^{(s)}} & (0 \leq t \wedge i \in I_{\mathrm{C}}), \\ \left\{1 - (y_{t,i}^{(s)})^2\right\} \dfrac{\hat{y}_{t,i}^{(s)} - y_{t,i}^{(s)}}{v_{t,i}^{(s)}} & (1 \leq t \wedge i \in I_{\mathrm{O}}), \\ -\dfrac{1}{2} + \dfrac{(\hat{y}_{t,i}^{(s)} - y_{t,i}^{(s)})^2}{2 v_{t,i}^{(s)}} & (1 \leq t \wedge i \in I_{\mathrm{V}}), \end{cases} \tag{2.22}$$

$$\frac{\partial \ln L_{\mathrm{all}}}{\partial u_{0,i}^{(s)}} = \frac{\partial \ln L_{\mathrm{out}}}{\partial u_{0,i}^{(s)}} - \frac{1}{\sigma_{\mathrm{IS}}^2} (u_{0,i}^{(s)} - u_i) \qquad (i \in I_{\mathrm{C}}), \tag{2.23}$$

$$\frac{\partial \ln L_{\mathrm{all}}}{\partial u_i} = \sum_{s \in I_{\mathrm{S}}} \frac{1}{\sigma_{\mathrm{IS}}^2} (u_{0,i}^{(s)} - u_i) \qquad (i \in I_{\mathrm{C}}). \tag{2.24}$$

2.7 Parameter Initialization for Predictive Learning

Synaptic weights w_{ij} were initialized with values randomly chosen from a uniform distribution on the intervals $\left[-\frac{1}{N_{\mathrm{I}}}, \frac{1}{N_{\mathrm{I}}}\right]$ (if $j \in I_{\mathrm{I}}$) and $\left[-\frac{1}{N_{\mathrm{C}}}, \frac{1}{N_{\mathrm{C}}}\right]$ (otherwise), where N_{I} and N_{C} are the numbers of the input and context units, respectively. Biases b_i were initialized with values randomly chosen from a uniform distribution on the interval $[-1, 1]$. Initial internal states $u_{0,i}^{(s)}$ and the mean values u_i of the initial states were set to 0 and values randomly chosen from a uniform distribution on the interval $\left[-\frac{1}{N_{\mathrm{C}}}, \frac{1}{N_{\mathrm{C}}}\right]$, respectively. Since the maximum value of L_{out} depends on the total length T_{total} of the target sequences and the dimensionality N_{O} of the output units, the learning rate α_{share} for updating weights and biases was scaled by a parameter $\tilde{\alpha}$ satisfying the relation $\alpha_{\mathrm{share}} = \frac{1}{T_{\mathrm{total}} N_{\mathrm{O}}} \tilde{\alpha}$. The learning rate for updating initial internal states and mean values was $\alpha_{\mathrm{init}} = \frac{1}{N_{\mathrm{O}}} \tilde{\alpha}$.

It should be noted that because most parameters are scaled by the features of training data such as the total length and the dimensionality of the data, the

above setting can be reused for other training data. Although the non-scaled parameters including the time constants and the numbers of the context units should be tuned by trial and error, learning results are not so sensitive to their setting.

2.8 Generation Method

After the predictive learning process, the S-CTRNN as a generative model is able to predict the future input state $\boldsymbol{x}^{(s)}_{t+\xi}$ from the current input state $\boldsymbol{x}^{(s)}_{t}$ given a specific initial internal state of context units ($\boldsymbol{u}^{(s)}_{0}$). This can be cast as the process to map a static causal state (initial state) to the corresponding dynamic consequence (temporal sequence) by means of forward dynamics.

There are the following two different ways to feed the current input $x^{(s)}_{t,i}$ into the network for generation:

$$x^{(s)}_{t,i} = \begin{cases} \hat{y}^{(s)}_{t-\xi,i}, & \text{(2.25a)} \\ y^{(s)}_{t-\xi,i} + \epsilon^{(s)}_{t-\xi,i}, & \text{(2.25b)} \end{cases}$$

where $\hat{y}^{(s)}_{t-\xi,i}$ is an external input state representing a recorded target state or actual sensory information acquired by a robot, $y^{(s)}_{t-\xi,i}$ is an output state or a predicted input state generated by the trained network, and $\epsilon^{(s)}_{t-\xi,i}$ is Gaussian noise given by

$$\epsilon^{(s)}_{t-\xi,i} = \epsilon(v^{(s)}_{t-\xi,i}) \sim \mathcal{N}(0, v^{(s)}_{t-\xi,i}), \tag{2.26}$$

where $\epsilon(\sigma^2)$ is a Gaussian noise generator with a zero mean and a standard deviation of σ (Fig. 2.4).

In (2.25a), the current input is the current external input, and this case is referred to as the *open-loop mode*. On the other hand, in (2.25b), the current input is derived from the predicted mean value to which Gaussian noise with the predicted variance at the previous step is added. This case is referred to as the *closed-loop mode* or more precisely, closed-loop mode with the addition of Gaussian noise with the predicted variance. Equation (2.25b) can autonomously generate fluctuating states whose ensembles are assumed to reproduce the stochastic structure latent in the target sequence. It should be noted that we can apply a different generation method for each output dimension. For example, it is possible to use the open-loop mode for visual states and the closed-loop mode with the

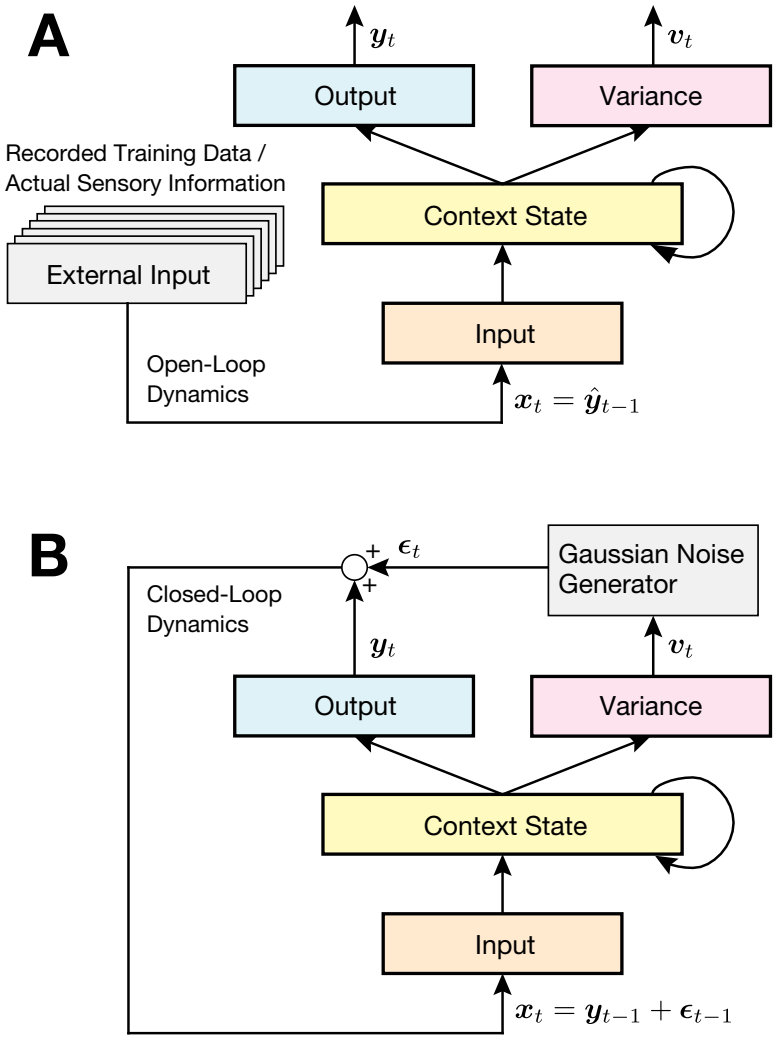


Figure 2.4 Generation method: (A) open-loop mode and (B) closed-loop mode with the addition of Gaussian noise with the predicted variance. For simplicity, an example with the delay length $\xi = 1$ is illustrated.

addition of Gaussian noise with the predicted variance for proprioceptive states for action generation by robots.

2.9 Recognition Method

In the preceding section, generation method using the S-CTRNN after the predictive learning is introduced as a mapping from a static causal state (initial internal state of context units) to its corresponding dynamic consequence (temporal sequence) with forward dynamics. The recognition of given temporal sequences can be performed as an inverse mapping, namely, the mapping from a dynamic consequence to the corresponding static causal state with backward dynamics. This means that S-CTRNNs as a generative model can be used for both the "generation" and "recognition" processes againt other approaches that require different models for each process, for example, *forward model* for generation and *inverse model* for recognition.

The recognition process based on such an inverse mapping is realized in a similar manner to the learnig process that is also based on the backward dynamics. The learning corresponds to the process of optimizing both the shared (weight and bias) and respective (initial state) parameters to generate or reproduce target sequences. On the other hand, the recognition corresponds to the process of inferring the optimized initial internal states of context units which can reproduce given sequences. This inference process is enabled by the gradient ascent method as well as the learning process. Although all the parameters (including the shared parameters collected in $\boldsymbol{\theta}_{\text{share}}$, which are common for all sequences, and the parameters regarding the initial internal states of context units collected in $\boldsymbol{\theta}_{\text{init}}$, which are different for each sequence) are optimized during the learning process, only the initial internal states $u^{(s)}_{0,i}$ are optimized during the recognition process. In other words, the five phases for the learning process mentioned in Section 2.6 can be reused only by changing the "parameters" in phases 1 and 4 to "only initial internal states of context units ($u^{(s)}_{0,i}$)".

2.10 Stochastic Multiple Timescale RNN (S-MTRNN)

This section considers an extention of the S-CTRNN by introducnig multiple timscale dynamics, which are implemented in the MTRNN introduced in the

preceding chapter (Section 1.4.2), to the context units. The extended model is called stochastic MTRNN (S-MTRNN).

Figure 2.5 shows a schematic illustration of the S-MTRNN. As shown in the figure, the context units of the S-MTRNN are divided into two groups characterized by a difference in time constants of neural activity. Hereinafter faster timescale units with a smaller time constant (τ_{FC}) are called fast context (FC) units, and slower timescale units with a larger time constant (τ_{SC}) are called slow context (SC) units. Yamashita and Tani [74] demonstrated that a difference of timescales in the MTRNN enables the self-organization of a functional hierarchy in which a set of action primitives can be stored in the FC units, and sequential combinations of the primitives can be represented in the SC units. In addition to the multiple timescales of the neural activity, the two groups of context units had different connectivities to introduce constraints on information flow. The FC units with the smaller time constant were connected with all units. On the other hand, the SC units with the larger time constant were only connected with the context units. In addition to the multiple timescale property, the constraint on information flow derived from the connection setting is also essential for the self-organization of functional hierarchy.

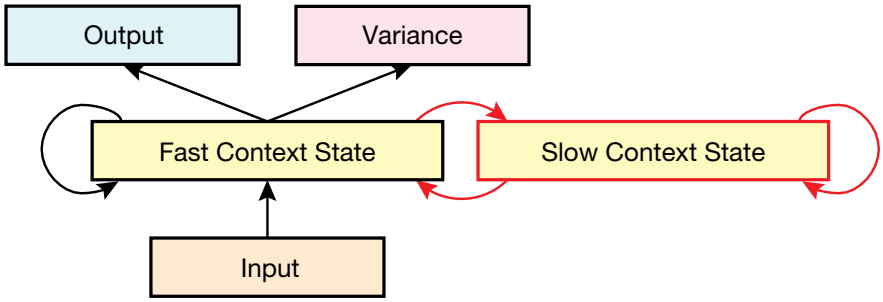


Figure 2.5 Schematic of S-MTRNN. The S-MTRNN consists of an input, a fast context (FC), a slow context (SC), an output, and a variance layer. The additional parts compared to the basic S-CTRNN shown in Fig. 2.2A are colored with red (the slow context states and the related connections). The other parts are the same as those for the S-CTRNN. The neural activity of the SC units with larger time constant is slow and that of the FC units with smaller time constant is fast. In addition to the difference of neural activity in the FC and SC units, these units have different connection settings. The FC units have connections from input, SC units to them, and from them to themselves, SC, output, and varaince units. The SC units have connections from FC units to them, and from them to FC units and themselves.

2.10.1 Forward Propagation

The forward dynamics of the internal states of the ith FC, SC, and output unit at time step t corresponding to the sth target sequence ($u_{t,i}^{(s)}$) are given by

$$u_{t,i}^{(s)} = \begin{cases} \left(1 - \frac{1}{\tau_i}\right) u_{t-1,i}^{(s)} + \frac{1}{\tau_i} \left\{ \left(\sum_{j \in I_{\mathrm{I}}} w_{ij} x_{t,j}^{(s)} \right) + \left(\sum_{j \in I_{\mathrm{FC}} \cup I_{\mathrm{SC}}} w_{ij} c_{t-1,j}^{(s)} \right) + b_i \right\} & (i \in I_{\mathrm{FC}} \cup I_{\mathrm{SC}}), \\ \left(\sum_{j \in I_{\mathrm{FC}}} w_{ij} c_{t,j}^{(s)} \right) + b_i & (i \in I_{\mathrm{O}} \cup I_{\mathrm{V}}), \end{cases} \tag{2.27}$$

where

- τ_i: time constant of the ith FC or SC unit (τ_{FC} or τ_{SC}),
- $I_{\mathrm{I}}, I_{\mathrm{FC}}, I_{\mathrm{SC}}, I_{\mathrm{O}}, I_{\mathrm{V}}$: index sets for the input, FC, SC, output, and variance units, respectively,
- w_{ij}: synaptic weight of the connection from the jth unit to the ith unit (if $i \in I_{\mathrm{SC}} \wedge j \in I_{\mathrm{I}}$ then $w_{ij} = 0$),
- $x_{t,j}^{(s)}$: the jth external input state at time step t corresponding to the sth targe sequence,
- $c_{t,j}^{(s)}$: neural activation state of the jth FC or SC unit at time step t corresponding to the sth target sequence,
- b_i: bias of the ith unit.

By considering $I_{\mathrm{C}} = I_{\mathrm{FC}} \cup I_{\mathrm{SC}}$, we can apply all the computation methods of the S-CTRNN descirbed before to those of the S-MTRNN.

The dynamics of the FC units started from a neutral initial state (zero value), and those of the SC units started from a particular initial state that had been optimized during the learning process. Thus, the two groups of FC and SC units can be regarded as a lower-level and a higher-level network, respectively. The higher level with SC units and the lower level with FC units may correspond to the rostral and the caudal part in cortex creating a so-called *rostro-caudal gradient* of timescales [90]. From the viewpoint of the integration of information processing and memory, Hasson et al. [91] proposed a hierarchical *process memory* framework based on their neurophysiological and neuroimaging studies. In their framework, the processing timescale of each area of cortex is characterized by the temporal receptive window (TRW), which is analogous to the time constants. They argue that the TRW gradually increases from early sensory areas to higher-order areas.

2.10.2 Dynamic Recognition Method

In a similar way to the recognition (more precisely, static recognition) of given temporal sequences described in Section 2.9, a dynamic recognition of situational changes can also be performed by inferring the internal state of the context units which can reproduce the target states within a certain time window of the immediate past. This is a novel error regression scheme (ERS) that is inspired by the ERS for the RNNPB introduced in the preceding chapter (Section. 1.4.1).

In the static recognition process, we consider the inference of the optimized initial internal states of context units ($u^{(s)}_{0,i}$) after observing (receiving) a whole sequence. On the other hand, in the dynamic recognition process, we consider the inference of the internal states of context units at the current time step t ($u^{(s)}_{t,i}$) by regressing immediate past states in the time window. Although the internal states of all the context units including FC and SC units can be inferred theoretically, only the inference of the internal state of SC units is considered for real-time computation. More specifically, the internal states of the SC units are modulated in a way that maximizes a part of the likelihood defined in (2.10) by regressing past states. Actually, because the lower-level network with FC units is influenced by the higher-level network with SC units, the neural activity of FC units can also be modulated by the inferred internal states of SC units. This is a formal extension of active inference [11] in the field of theoretical neurobiology.

The internal states of the SC units at time step $t-W$ ($u^{(s)}_{t-W,i}$) are dynamically modulated by using the gradient ascent method to maximize the likelihood

$$L_{\mathrm{reg}} = \prod_{t'=t-W+1}^{t} \prod_{i \in I_{\mathrm{O}}} \frac{1}{\sqrt{2\pi v^{(s)}_{t',i}}} \exp\left(-\frac{(\hat{y}^{(s)}_{t',i} - y^{(s)}_{t',i})^2}{2v^{(s)}_{t',i}} \right), \tag{2.28}$$

where the same BPTT scheme [83] adopted for the learning and static recognition process was used without changing the synaptic weights and biases, and W is the length of the time window that shifts along with the increment of the time step t'. This error regression was conducted for several regression steps for each time step t. In this study, the mean and variance predictions about target states and the context states for time steps from $t-W+1$ to t were generated using closed-loop generation (2.25b) without adding noise in which the "re-interpreted" or "postdicted" [92] mean state was used as an input for the next time step. During this generation mode, the set of internal states of the context units at time step $t-W$ serves as an initial internal state within the time window for ERS. Although

we can consider a balance between the forward dynamics state predicted in the past and that state postdicted at the present with the regression, in the present study the scheme assumed that the former state is completely overwritten with the latter state. The robot experiment described in Chapter 5 investigates the difference of behavior produced by the robot driven by the trained S-MTRNN with or without dynamic recognition using the ERS.

Chapter 3

Predictive Learning of Fluctuating Temporal Sequences

3.1 Introduction

This chapter focuses on the numerical evaluation of the proposed computational framework for predictive learning with uncertainty estimation introduced in the preceding chapter. Three numerical experiments are conducted to demonstrate how the S-CTRNN based on the proposed framework learns, reproduces, and recognizes fluctuating or noisy temporal sequence data. Each experiment employs different sort of temporal sequence data including trajectory-level uncertainty for the predictive learning which are artificially generated by adding stochastic or random fluctuations to clean target states or to a dynamical system generating these states.

The first experiment, described in Section 3.2, considers the comparison of learning capabilities between the conventional CTRNN and the proposed S-CTRNN. This experiment involves learning a set of temporal sequences (multiple Lissajous curves) with random fluctuations. The fluctuations are derived from the added Gaussian noise whose variance is both sequence-specific and time-invariant. The results demonstrate that the conventional CTRNN fails to learn these fluctuating temporal sequences. The S-CTRNN, in contrast, can learn all the temporal sequences as multiple limit cycle attractors and can also reproduce the stochastic structures (sequence-specific variances) latent in the data. The results of recognition of given sequences using the trained S-CTRNN are also demonstrated. The second experiment, described in Section 3.3, considers the extraction and reproduction of another type of stochastic structures. This experiment involves

learning a temporal sequence (sinusoidal curve) with Gaussian noise whose variance changes temporally with a certain period. The results demonstrate that the S-CTRNN can extract and reproduce such a time-varying stochastic structure via the predictive learning with uncertainty estimation.

In the above-mentioned experiments, Gaussian noise is added to the reference trajectories of the Lissajous curves or a sinusoidal curve at each time step. Therefore, the generated (or observed) target states exhibit discontinuous fluctuations. The third experiment, described in Section 3.4, considers the extraction and reproduction of the other type of stochastic structures. This experiment involves learning temporal sequences that are generated from a random dynamical system (limit cycle attractor) in which Gaussian noise with state-dependent variance is added. Because Gaussian noise is added not to reference trajectories but to the dynamical system itself, the stochastic structure of the generated target states is relatively more continuous than the previous cases. The results demonstrate that the S-CTRNN can also extract and reproduce such a state-dependent stochastic structure via the predictive learning with uncertainty estimation.

3.2 Learning of Multiple Fluctuating Lissajous Curves

This section describes the results of learning, reproduction, and recognition of fluctuating temporal sequences by extracting multiple time-invariant uncertainty in terms of variance prediction. In the experiment, fluctuating temporal sequences for predictive learning were produced by adding Gaussian noise with sequence-specific and time-invariant variance to clean target states (or reference trajectories).

3.2.1 Training Data

Training data $\{\hat{\boldsymbol{y}}_t^{(s)}\}_{t=1}^{T^{(s)}}$ for predictive learning consisted of 12 Lissajous curves with random fluctuations ($s \in I_{\rm S} = \{1, 2, \cdots, 12\}$). To prepare the training data, clean Lissajous curves with a period $P = 25$ and a length $T^{(s)} = 1000$ $(= 40P)$ were first generated as combinations of the following four sinusoidal curves $\hat{Y}_{t,i}$

($i \in \{1, 2, 3, 4\}$):

$$\begin{cases} \hat{Y}_{t,1} = -0.4 \left\{ \cos\left(\dfrac{2\pi t}{P}\right) - 1 \right\}, \\ \hat{Y}_{t,2} = 0.8 \sin\left(\dfrac{2\pi t}{P}\right), \\ \hat{Y}_{t,3} = -0.4 \left\{ \cos\left(\dfrac{4\pi t}{P}\right) - 1 \right\}, \\ \hat{Y}_{t,4} = 0.8 \sin\left(\dfrac{4\pi t}{P}\right). \end{cases} \tag{3.1}$$

Then, Gaussian noise $\hat{\epsilon}_{t,i}^{(s)}$ ($i \in I_{\mathrm{O}} = \{1, 2\}$) with sequence-specific and time-invariant variance $\hat{v}_t^{(s)} = \{\hat{\sigma}^{(s)}\}^2$ was added to each dimension of the clean Lissajous curves at each time step. The resultant fluctuating target states $(\hat{y}_{t,1}^{(s)}, \hat{y}_{t,2}^{(s)})$ were expressed as follows (refer to Appendix C for details about all expressions):

$$\begin{aligned} (\hat{y}_{t,1}^{(1)}, \hat{y}_{t,2}^{(1)}) &= (\hat{Y}_{t,1} + \hat{\epsilon}_{t,1}^{(1)}, \hat{Y}_{t,2} + \hat{\epsilon}_{t,2}^{(1)}), \\ &\vdots \\ (\hat{y}_{t,1}^{(12)}, \hat{y}_{t,2}^{(12)}) &= (\hat{Y}_{t,2} + \hat{\epsilon}_{t,1}^{(12)}, -\hat{Y}_{t,3} + \hat{\epsilon}_{t,2}^{(12)}). \end{aligned} \tag{3.2}$$

The value of the standard deviation, which is the square root of the sequence-specific and time-invariant variance, of the added Gaussian noise was defined as

$$\hat{\sigma}^{(s)} = \begin{cases} 0.01 & (s \in \{1, 5, 9\}), \\ 0.03 & (s \in \{2, 6, 10\}), \\ 0.05 & (s \in \{3, 7, 11\}), \\ 0.07 & (s \in \{4, 8, 12\}). \end{cases} \tag{3.3}$$

Figure 3.1 presents phase plots of the 12 fluctuating Lissajous curves. The predictive learning task used a set of these 12 fluctuating temporal sequences with the length $T^{(s)} = 1000$ as training data including multiple time-invariant uncertainty. It was considered that the fluctuating temporal sequences can be embedded as multiple limit cycle attractors in a single S-CTRNN by self-organizing the corresponding initial internal states of context units.

3.2.2 Parameter Settings for Predictive Learning

Both the conventional CTRNN and the proposed S-CTRNN were trained with the same parameter settings for comparison of learning capabilities. The number

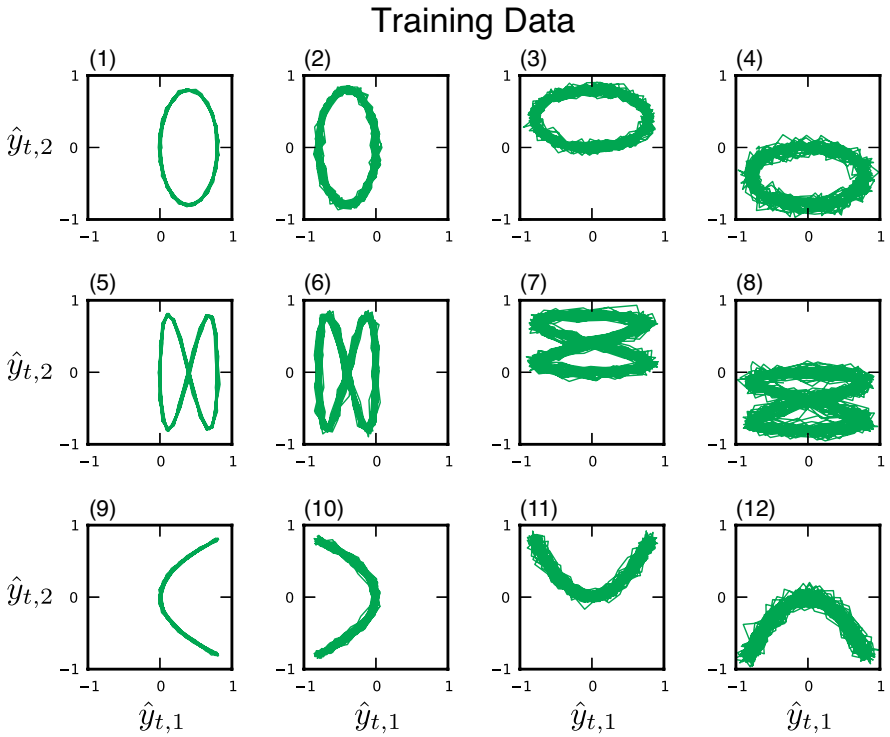


Figure 3.1 Twelve fluctuating Lissajous curves in training data. Gaussian noise $\hat{\epsilon}_{t,i}^{(s)} \sim \mathcal{N}(0, \hat{v}^{(s)})$ whose variance $\hat{v}^{(s)}$ depends on the sequence s was added in $\hat{y}_{t,1}^{(s)}$ and $\hat{y}_{t,2}^{(s)}$. In each row, the variance increases from left to right ($\hat{v}^{(s)} = 0.0001, 0.0009, 0.0025, 0.0049$).

of input, output, and variance (only for the S-CTRNN) units were $N_{\mathrm{I}} = N_{\mathrm{O}} = N_{\mathrm{V}} = 2$, respectively. These were determined by the dimensionality of the target sequences. The delay length that determines the relationship between input states and target states was set to $\xi = 1$, namely, $\boldsymbol{x}_t^{(s)} = \hat{\boldsymbol{y}}_{t-1}^{(s)}$. The number of context units, the time constant of the context units, and the variance of the initial internal states of the context units were chosen to be $N_{\mathrm{C}} = 60$, $\tau_i = 2$ ($i \in I_{\mathrm{C}} = \{1, 2, \cdots, N_{\mathrm{C}}\}$), and $\sigma_{\mathrm{IS}}^2 = 100$, respectively. Here, in order to embed multiple attractors into a single network, a large value was chosen for the variance of the initial states. The parameter for determining the learning rate was chosen to be $\tilde{\alpha} = 0.01$ for the CTRNN and $\tilde{\alpha} = 0.0001$ for the S-CTRNN. This difference was attributed to the difference of the objective function of each network, namely, the prediction error for the CTRNN and the model likelihood for the S-CTRNN in which prediction error is scaled by the predicted variance. The momentum term was chosen to be $\eta = 0.9$.

The network parameters optimized in the learning process including synaptic weights, biases, and initial internal states of context units were initialized in the way written in Chapter 2 (Section 2.7).

3.2.3 Comparison of Learning Capabilities between CTRNN and S-CTRNN

Both the conventional CTRNN and the proposed S-CTRNN were trained for 500,000 training epochs by optimizing the network parameters with the open-loop mode (2.25a). After the training, the trained networks were evaluated whether they were able to reproduce all the learned Lissajous curves by using the respective optimized initial internal state for each network.

Temporal sequences of output states were generated from the CTRNN ($y_{t,1}^{(s)}, y_{t,2}^{(s)}$) and from the S-CTRNN ($y_{t,1}^{(s)} + \epsilon(v_{t,1}^{(s)}), y_{t,2}^{(s)} + \epsilon(v_{t,2}^{(s)})$) with the closed-loop dynamics by setting each optimized initial internal state $u_{0,i}^{(s)}$ ($i \in I_{\mathrm{C}}$). Especially in the case of the S-CTRNN, Gaussian noise with the predicted variance $v_{t,i}^{(s)}$ ($i \in I_{\mathrm{V}}$) was added during the closed-loop generation by using (2.25b). Figure 3.2 presents phase plots of output states of the CTRNN, and output states (with the addition of Gaussian noise) of the S-CTRNN, respectively. By comparing both sets of the phase plots of the training data (Fig. 3.1) and the output states (Fig. 3.2A), we can see that the CTRNN failed to generate some patterns whose noise variance are small (refer to Appendix C for learning results in different random seed cases).

Specifically for example, the three patterns with the smallest noise variance (1, 5, 9) are corrupted by the three patterns with the largest noise variance (4, 12, 8). The network also failed to reproduce the stochastic structures latent in the target sequences.

In contrast, the S-CTRNN succeeded to reproduce the target sequences with the corresponding stochastic structures (Fig. 3.2B). 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 limit cycle attractors. Figure 3.3 presents phase plots of two selected context activation states of the S-CTRNN. It appears that the context states lack stochastic structures corresponding to the target sequences although output states show them. This finding is revisited in the discussion.

3.2.4 Extraction of Multiple Time-Invariant Uncertainty

The temporal sequences of the variance states predicted by the trained S-CTRNN and their true values for each pattern are shown in Fig. 3.4. Although the values of the predicted variance $v_{t,1}^{(s)}$ oscillate, we can see that they are close to the true value $\hat{v}^{(s)}$.

In order to evaluate the accuracy of the predicted variance, the following temporal mean of predicted variance ($\overline{v}_i^{(s)}$) of the trained network for each sequence s was computed,

$$\overline{v}_i^{(s)} = \frac{1}{T^{(s)}} \sum_{t=1}^{T^{(s)}} v_{t,i}^{(s)}, \tag{3.4}$$

where $T^{(s)} = 1000$ is the length of the sequence, as mentioned above. Table 3.1 shows the calculated mean and standard deviation (SD) values of the predicted variances $\overline{v}_i^{(s)}$ and their corresponding true values $\hat{v}^{(s)}$.

3.2.5 Recognition Results

In a recognition experiment, the trained S-CTRNN was evaluated whether it was able to infer the initial internal states of the context units given multiple temporal patterns. Although all parameters (including the synaptic weights, biases, and initial internal states) were updated 500,000 training epochs during the predictive learning, the initial internal states were updated only 300 epochs, with the remaining parameters fixed during the recognition process as explained

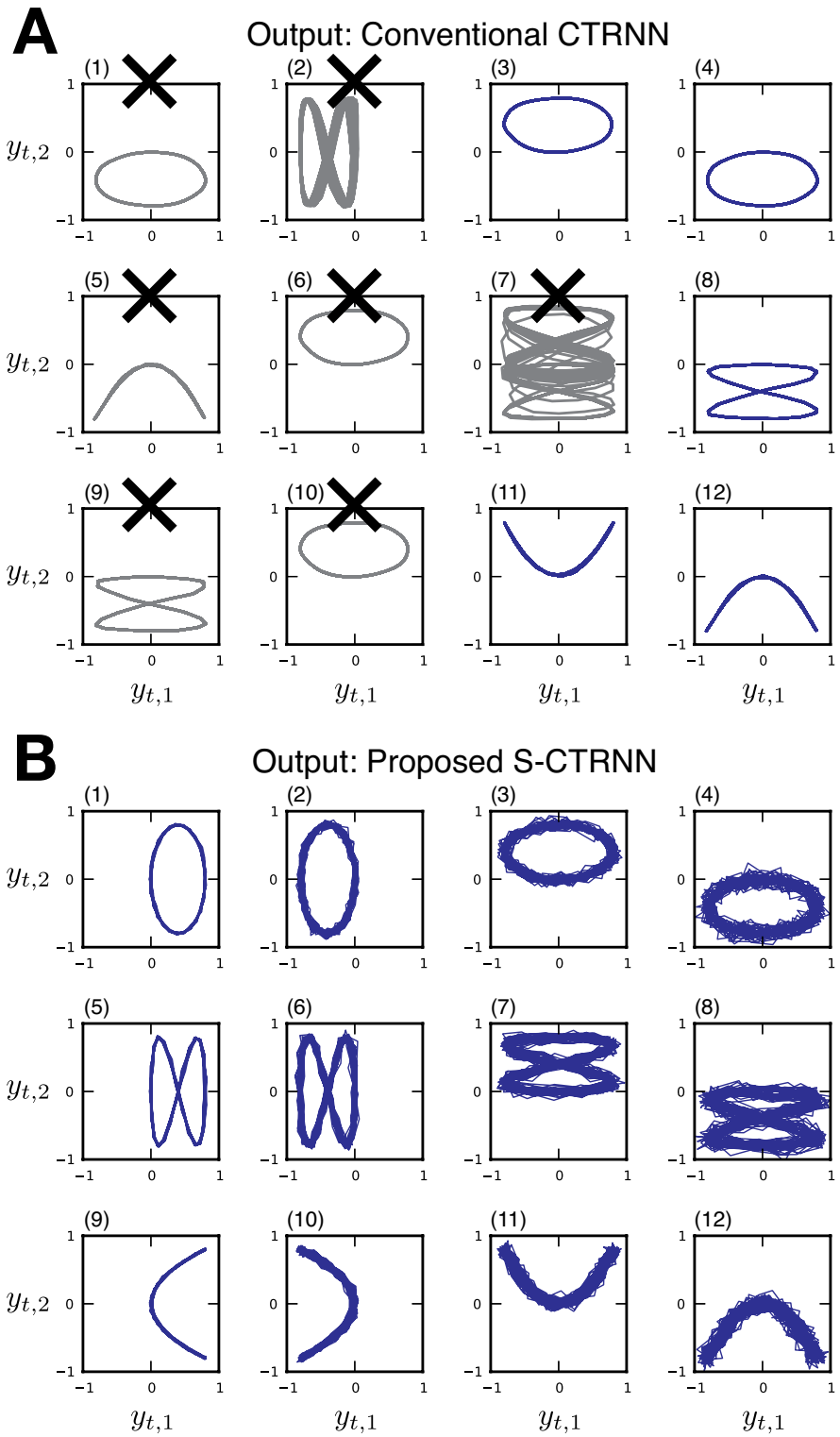


Figure 3.2 Phase plots of output states generated by the trained network: (A) output states of the trained CTRNN with closed-loop dynamics, and (B) output states of the trained S-CTRNN with closed-loop dynamics with the addition of Gaussian noise with the predicted variance. The crosses mark cases of failure.

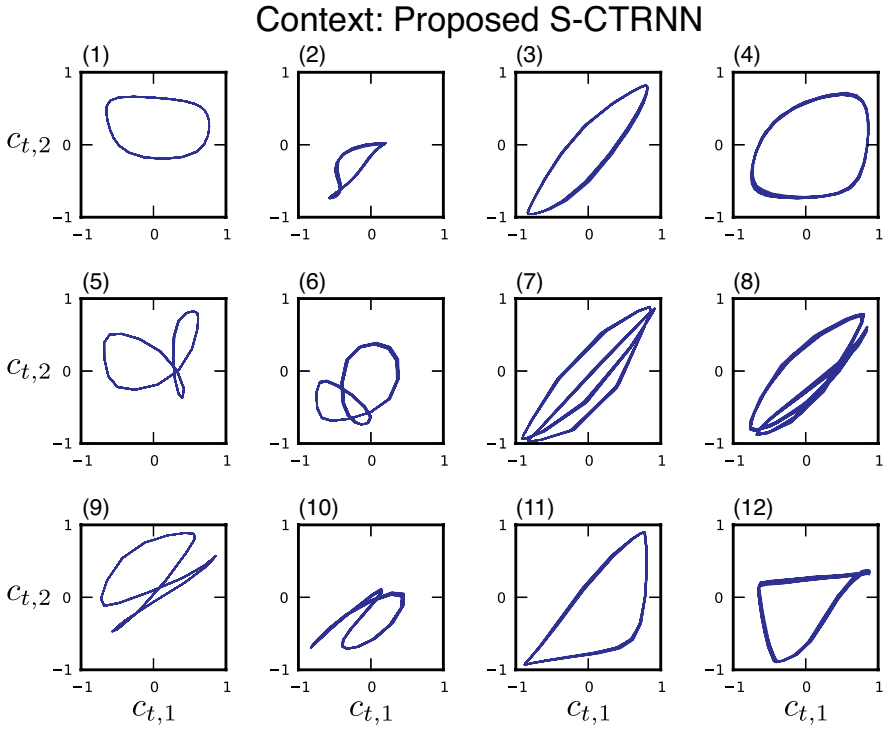


Figure 3.3 Phase plots of two selected context activation states of the trained S-CTRNN. There are no fluctuations observed in output states of the trained S-CTRNN shown in Fig. 3.2.

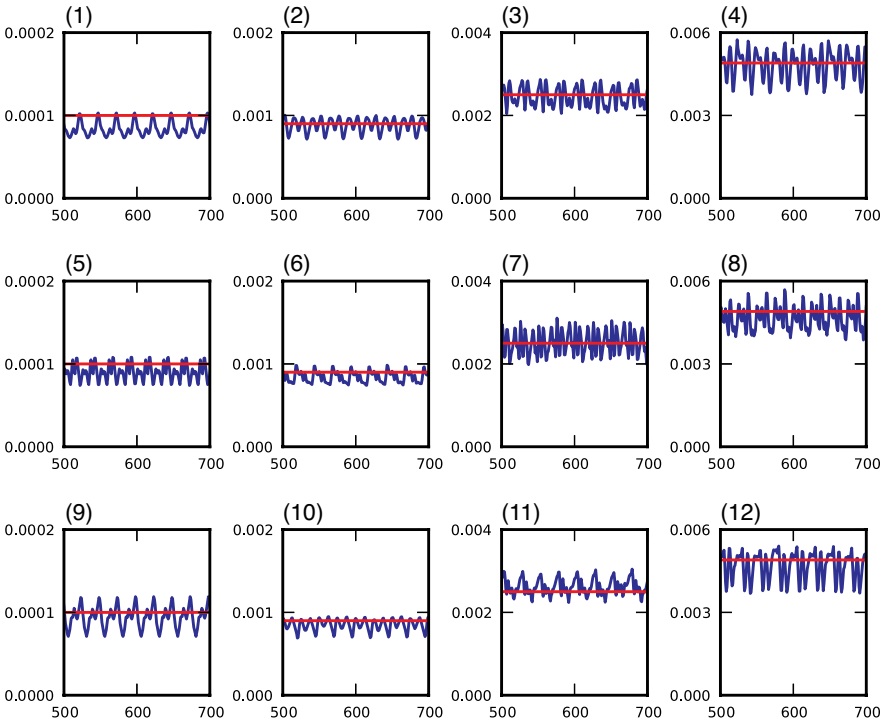


Figure 3.4 Temporal sequences of the predicted variances of the trained S-CTRNN in the case of closed-loop dynamics with the addition of Gaussian noise with the predicted variance for time steps 500–700. The blue line indicates the predicted variance $v_{t,1}^{(s)}$ of the trained network, and the red line indicates its true value $\hat{v}^{(s)}$.

Table 3.1 Comparison between mean of predicted variances and the corresponding true values.

Index of Sequence	**True Value**	**Mean and SD of Predicted Variance**			
s	$\hat{v}^{(s)}$	$\overline{v}_1^{(s)}$	$\mathrm{SD}_1^{(\mathrm{s})}$	$\overline{v}_2^{(s)}$	$\mathrm{SD}_2^{(\mathrm{s})}$
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

in Chapter 2 (Section 2.9). In the experiment, the search for initial states using the gradient ascent method sometimes falls to a local maximum. Therefore, 20 different sets of randomly chosen values for the initial internal state for each sequence were tested.

In order to evaluate the recognition error, the following mean square error (MSE) for each sequence s was computed:

$$E^{(s)} = \frac{1}{2T^{(s)}N_{\mathrm{O}}} \sum_{t=1}^{T^{(s)}} \sum_{i=1}^{N_{\mathrm{O}}} (\hat{y}_{t,i}^{(s)} - y_{t,i}^{(s)})^2, \tag{3.5}$$

where $\hat{y}_{t,i}^{(s)}$ and $y_{t,i}^{(s)}$ are the ideal value in the target states and the output value of the S-CTRNN corresponding to the sth sequence, respectively. $T^{(s)} = 1000$ and $N_{\mathrm{O}} = 2$ are the length and output dimension of each sequence, respectively. As a criterion for the success or failure of recognition, the upper bound of MSE for successful recognition was set to $(3\hat{\sigma}^{(s)})^2/2$ with respect to the noise variance of each trained pattern.

Table 3.2 shows the total number of successful recognitions out of 20 trials. All fluctuating patterns were recognized, although the success rate for the patterns with the largest noise variance (4, 8, 12) was lower than that for the patterns with the smallest noise variance (1, 5, 9).

Table 3.2 Number of successful recognitions (SR) out of 20 trials.

Sequence	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
Number of SR	17	14	11	9	11	16	9	1	12	16	9	9

3.2.6 Initial State Space Analysis

The 60-dimensional initial internal state space of the context units, which was self-organized during the learning process, was visualized by applying principal component analysis (PCA) in the space. Figure 3.5 shows the compressed initial state space defined by the first and second principal components, where we can see four clusters based on the noise variance of the training patterns.

3.3 Learning of Fluctuating Sinusoidal Curve

This section describes the results of learning and reproduction of a fluctuating temporal sequence by extracting time-varying uncertainty in terms of variance

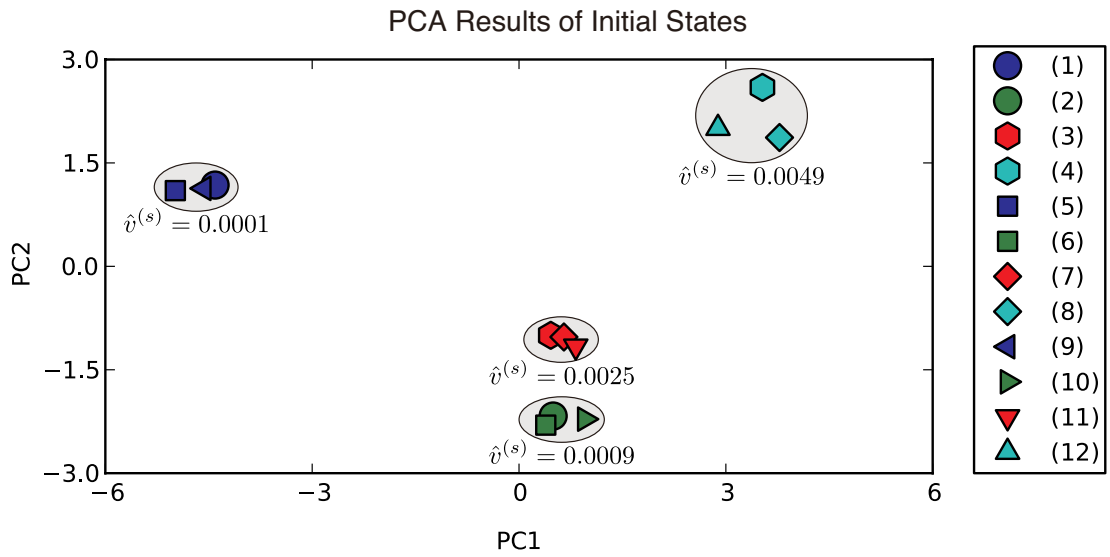


Figure 3.5 Initial internal state space of the context units. The dimensionality of the space was reduced from 60 to 2 by PCA. Four clusters based on the noise variance of the training patterns can be observed.

prediction. In the experiment, a fluctuating temporal sequence was produced by adding Gaussian noise with time-varying variance to clean target states (or reference trajectories).

3.3.1 Training Data

Training data $\{\hat{y}_t\}_{t=1}^{T}$ for predictive learning consisted of a one-dimensional fluctuating sinusoidal curve with a period $P = 25$ and a length $T = 1000$ $(= 40P)$. Gaussian noise $\hat{\epsilon}_t$ with time-varying variance $\hat{v}_t = \hat{\sigma}_t^2$ was added to the clean sinusoidal curve at each time step as follows:

$$\hat{y}_t = 0.8 \sin\left(\frac{2\pi t}{P}\right) + \hat{\epsilon}_t. \tag{3.6}$$

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

$$\hat{\sigma}_t = \begin{cases} \dfrac{0.12}{P}(t - nP) + 0.01 & \left(nP \le t < \dfrac{(2n+1)}{2}P\right), \\ -\dfrac{0.12}{P}(t - (n+1)P) + 0.01 & \left(\dfrac{(2n+1)}{2}P \le t < (n+1)P\right), \end{cases} \tag{3.7}$$

where n is a non-negative integer ($n \in \{0, 1, 2, \cdots, 39\}$). The predictive learning task used one target sequence with the length $T = 1000$ as training data including time-varying uncertainty.

3.3.2 Parameter Settings for Predictive Learning

The number of input, output, and variance units were $N_{\mathrm{I}} = N_{\mathrm{O}} = N_{\mathrm{V}} = 1$, respectively. These were determined by the dimensionality of the target sequence. The delay length that determines the relationship between input states and target states was set to $\xi = 1$, namely, $x_t = \hat{y}_{t-1}$. The number of context units, the time constant of the context units, and the variance of the initial internal states of the context units were chosen to be $N_{\mathrm{C}} = 10$, $\tau_i = 2$ ($i \in I_{\mathrm{C}} = \{1, 2, \cdots, N_{\mathrm{C}}\}$), and $\sigma^2_{\mathrm{IS}} = 1$, respectively. The parameter for determining the learning rate and the momentum term were chosen to be $\tilde{\alpha} = 0.0001$ and $\eta = 0.9$, respectively.

The network parameters optimized in the learning process including synaptic weights, biases, and initial internal states of context units were initialized in the way written in Chapter 2 (Section 2.7).

3.3.3 Extraction of Time-Varying Uncertainty

The S-CTRNN was trained for 100,000 training epochs with the open-loop mode (2.25a). After the training, the trained network was evaluated whether it was able to reproduce the learned pattern by using the optimized initial internal state. Temporal sequences ($y_t + \epsilon(v_t)$) were generated with the closed-loop dynamics with the addition of Gaussian noise with the predicted variance by using (2.25b). An example of generated temporal sequences is shown in Fig. 3.6. As can be seen from the figure, the output states of the trained network reproduces the stochastic structure latent in the target sequence, which consists of a periodic pattern with a time-varying variance. Furthermore, the predicted variance of the trained network v_t is close to the true value $\hat{v}_t = \hat{\sigma}^2_t$.

3.4 Learning of Random Dynamical System

This section describes the results of learning and reproduction of fluctuating temporal sequences by extracting state-deponent uncertainty in terms of variance prediction. In the experiment, fluctuating temporal sequences were produced by adding Gaussian noise with state-dependent variance not to target states but to

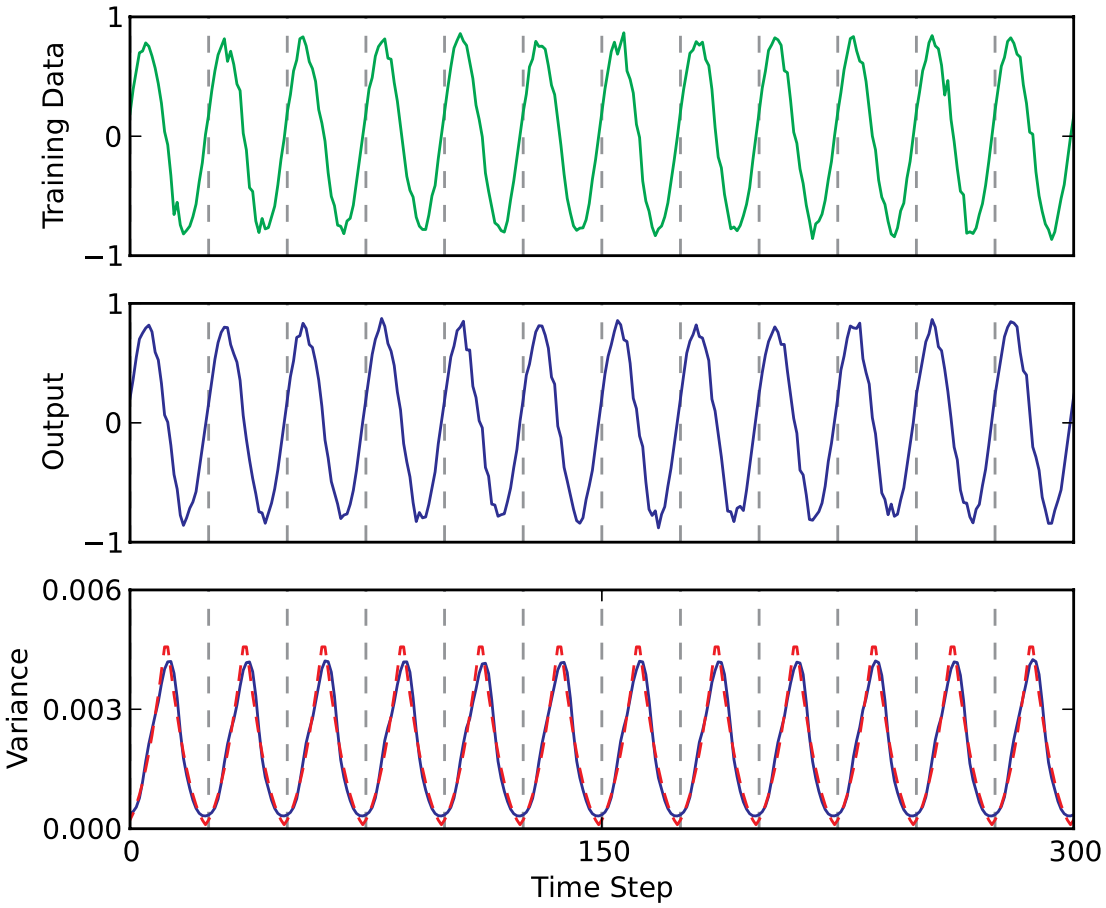


Figure 3.6 Temporal sequences with time-varying uncertainty. Training data (first row), output of the trained S-CTRNN with closed-loop dynamics with the addition of Gaussian noise with the predicted variance (second row), and variance predicted by the trained network (third row). In the third row (temporal sequences of variance), the blue line indicates the variance v_t predicted by the trained network, and the red dashed line indicates its true value $\hat{v}_t$.

a dynamical system generating these states, such that the noise can affect not only the current but also subsequent states.

3.4.1 Training Data

Training data $\{\hat{y}_t^{(s)}\}_{t=1}^{T^{(s)}}$ for predictive learning were generated by using the following piecewise stochastic differential equations:

$$\begin{cases} d\hat{y}(t) = z(t)dt + \sigma(\hat{y}(t))dB(t), \\ dz(t) = (f(z(t)) - k_1\hat{y}(t))dt. \end{cases} \tag{3.8}$$

$$f(z(t)) = \begin{cases} k_2 z(t) & (|z(t)| \leq z_a), \\ -k_2(z(t) - 2z_a) & (z_a < z(t)), \\ -k_2(z(t) + 2z_a) & (z(t) < -z_a), \end{cases} \tag{3.9}$$

where $B(t)$ denotes Brownian motion. In fact, target sequences were computed according to the following equations, which are numerical approximations of (3.8) and (3.9):

$$\begin{cases} \hat{y}_t = \hat{y}_{t-1} + z_{t-1}\Delta t + \hat{\epsilon}_t, \\ z_t = z_{t-1} + (f(z_{t-1}) - k_1\hat{y}_{t-1})\Delta t. \end{cases} \tag{3.10}$$

$$f(z_{t-1}) = \begin{cases} k_2 z_{t-1} & (|z_{t-1}| \leq z_a), \\ -k_2(z_{t-1} - 2z_a) & (z_a < z_{t-1}), \\ -k_2(z_{t-1} + 2z_a) & (z_{t-1} < -z_a). \end{cases} \tag{3.11}$$

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

$$\hat{\sigma}_t = \begin{cases} 0.03\hat{y}_{t-1} & (0 < \hat{y}_{t-1}), \\ 0 & (\text{else}). \end{cases} \tag{3.12}$$

The predictive learning task used a set of 10 target sequences generated from the above equations, each with a length $T^{(s)} = 1000$ as training data including state-dependent uncertainty. Here, only one-dimensional sequences $\hat{y}_t^{(s)}$ were used for the learning ($z_t^{(s)}$ was used for calculation of $\hat{y}_t^{(s)}$).

3.4.2 Parameter Settings for Predictive Learning

The number of input, output, and variance units were $N_{\mathrm{I}} = N_{\mathrm{O}} = N_{\mathrm{V}} = 1$, respectively. These were determined by the dimensionality of the target sequences. The delay length that determines the relationship between input states and target states was set to $\xi = 1$, namely, $x_t^{(s)} = \hat{y}_{t-1}^{(s)}$. The number of context units, the time constant of the context units, and the variance of the initial internal states of the context units were chosen to be $N_{\mathrm{C}} = 10$, $\tau_i = 2$ ($i \in I_{\mathrm{C}} = \{1, 2, \cdots, N_{\mathrm{C}}\}$), and $\sigma_{\mathrm{IS}}^2 = 1$, respectively. The parameter for determining the learning rate and the momentum term were chosen to be $\tilde{\alpha} = 0.0001$ and $\eta = 0.9$, respectively.

The network parameters optimized in the learning process including synaptic weights, biases, and initial internal states of context units were initialized in the way written in Chapter 2 (Section 2.7).

3.4.3 Extraction of State-Dependent Uncertainty

The S-CTRNN was trained for 500,000 training epochs with the open-loop mode (2.25a). After the training, the trained network was evaluated whether it was able to reproduce the learned pattern by using the optimized initial internal state. Temporal sequences ($y_t + \epsilon(v_t)$) were generated with the closed-loop dynamics with the addition of Gaussian noise with the predicted variance by using (2.25b). An example of generated temporal sequences is shown in Fig. 3.7. As can be seen from the figure, the output states of the trained network reproduces the stochastic structure latent in the target sequences, 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 $\hat{v}_t = \hat{\sigma}_t^2 = (0.03 y_{t-1})^2$.

Figure 3.8 presents phase plots of training data $\hat{y}_t - z_t$, the output of the trained network $y_t - z_t$, and the two selected context activation states. Comparing the phase plots of the training data and the output generated by the closed-loop dynamics, we can see that the trained network can generate a dynamic stochastic structure similar to that of the target sequence. Furthermore, it appears that the context state also shares the same dynamic stochastic structure.

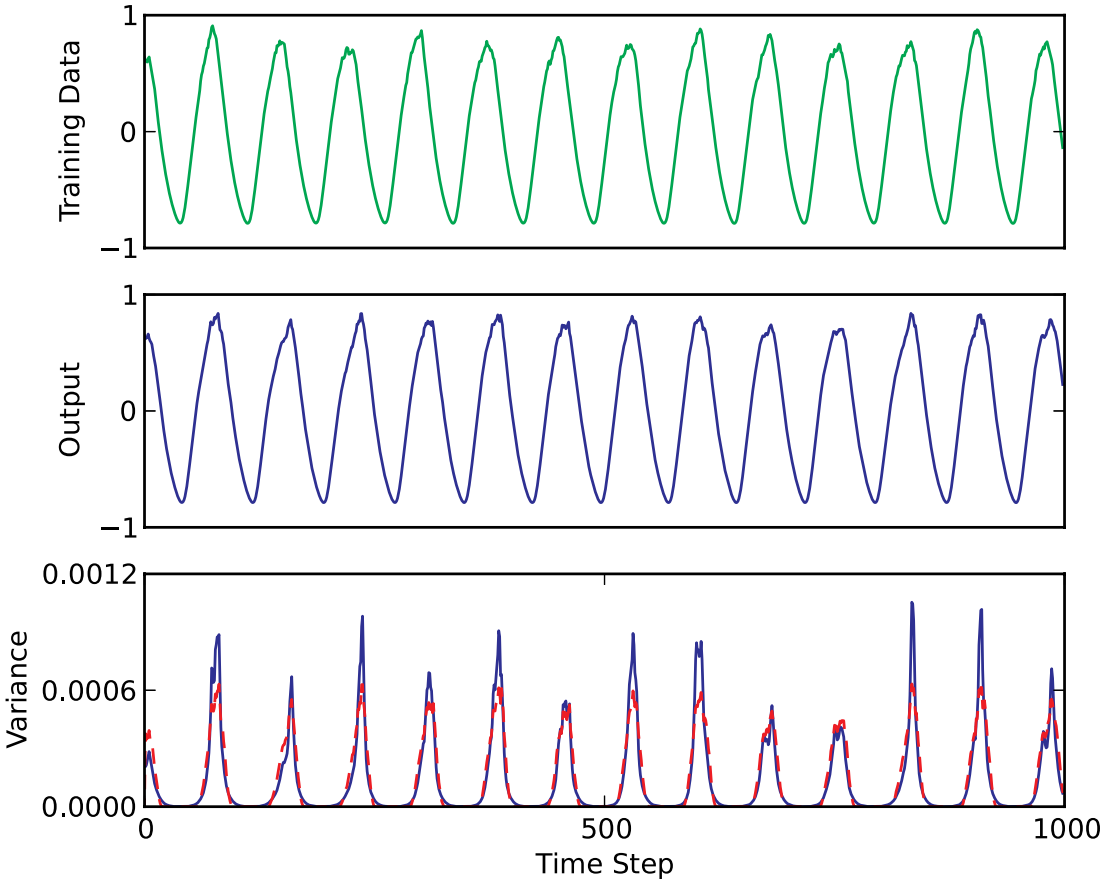


Figure 3.7 Temporal sequences with state-dependent uncertainty. Training data (first row), output of the trained S-CTRNN with the addition of Gaussian noise with the predicted variance (second row), and variance predicted by the trained network (third row). In the third row (temporal sequences of variance), 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 (3.12) as follows: $\hat{v}_{t+1} = (\hat{\sigma}_{t+1})^2 = (0.03y_t)^2$.

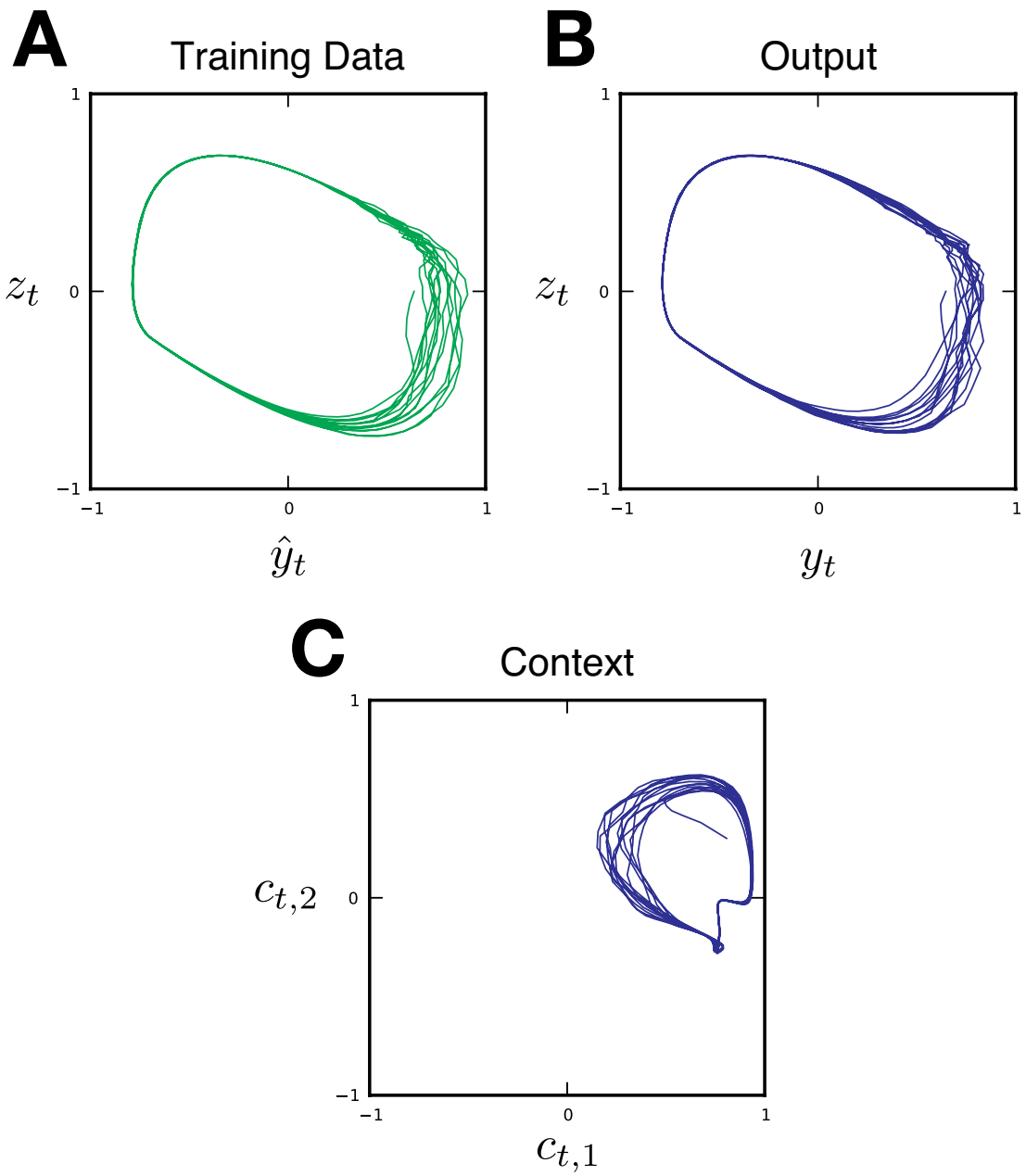


Figure 3.8 Comparison of phase plots: (A) training data $\hat{y}_t - z_t$, (B) output of the trained S-CTRNN with closed-loop dynamics with the addition of Gaussian noise with the predicted variance $y_t - z_t$, and (C) two selected context activation states of the trained network.

3.5 Discussion and Conclusions

In the first experiment described in Section 3.2, the task was predictive learning of the multiple fluctuating Lissajous curves in which Gaussian noise with sequence-specific and time-invariant variance was added. The experimental results revealed that the conventional CTRNN fails to learn these fluctuating temporal sequences (see Fig. 3.2A). Training of CTRNNs whose parameters were initialized with different random values showed that (1) patterns with smaller uncertainty were corrupted by patterns with larger uncertainty, (2) untrained attractors appeared, and (3) almost none of the training patterns were learned (see Fig. C.1). These results imply that when a CTRNN is used, temporal sequences with large uncertainty (or large fluctuations) tend to form strong attractors with a wide basin that attract a board range of states due to their large prediction error. Therefore, temporal sequences with smaller uncertainty (or smaller fluctuations) are corrupted by the former, to the extent where in some cases almost none of the target sequences are learned because this corruption mechanism interferes with the entire predictive learning process.

These problems can be solved by using an S-CTRNN in which the predicted variance scales the prediction error (refer to (2.12)) that contributes to the formation of attractors. The S-CTRNN can learn the temporal sequences as multiple limit cycle attractors with multiple time-invariant uncertainty (see Fig. 3.2B). The variance prediction mechanism also enables the network both to reproduce the stochastic structures of the fluctuating temporal sequences by adding Gaussian noise with the predicted variance in the closed-loop dynamics and to recognize these sequences by adequately scaling the prediction error used for the BPTT process in the same manner as the learning process. This sequence-specific variance prediction is enabled by the self-organized initial state of the context units whose space represents isolated clusters based on the noise variance of the training data (see Fig. 3.5).

In the second experiment described in Section 3.3, the task was predictive learning of the fluctuating sinusoidal curve in which Gaussian noise with the time-varying variance was added. It is confirmed that the variance predicted by the S-CTRNN is close to the correct one (see Fig. 3.6). It should be noted that in both of the above-mentioned experiments, Gaussian noise was added not to the intrinsic dynamics of generating those patterns but directly to the reference trajectories of the target sequences at each time step.

In contrast, in the third experiment described in Section 3.4, Gaussian noise was added to the dynamics of the limit cycle attractor itself. In this case as well, the S-CTRNN was able to predict the variance correctly (see Fig. 3.7) and to reproduce the stochastic structure hidden in the training data (see the training data and the output in Fig. 3.8). By comparing the context states in Figs. 3.3 and 3.8C, 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 3.2 and 3.3, 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 Fig. 3.3). On the other hand, in the case where noise was added to the dynamical system generating the target states, such as in the case described in Section 3.4, the stochastic structure appeared in the context trajectories in the trained network. Thanks to the fluctuations stored in the contextual dynamics, because noise added at previous time steps affects future states, relatively continuous fluctuating output states can be generated (see Fig. 3.8C).

Chapter 4

Predictive Learning to Develop Adaptive Behavior

4.1 Introduction

In the preceding chapter, the proposed S-CTRNN is evaluated in terms of its learning, reproduction, and recognition capabilities through a set of simple numerical experiments employing different sort of fluctuating temporal sequences with one or two dimensions. This chapter considers more realistic situations by applying the S-CTRNN into a framework for integrative learning of exteroceptive (or visual) states and proprioceptive states of a humanoid robot. Namely, the S-CTRNN is utilized as a generative model for learning to generate predictions about visuo-proprioceptive states.

The experiment focuses on the development of adaptive behavior by extracting stochastic structures latent in fluctuating behavioral (or visuo-proprioceptive) sequences with trajectory-level uncertainty. In the experiment, a small humanoid robot equipped with the S-CTRNN is required to learn a primitive skill for reaching movement toward a fixed target object from multiple demonstrations by means of kinesthetic teaching, where the robot's arm is directly guided by a human tutor. A resultant ensemble of proprioceptive states in behavioral trajectories represents stochastic structures such that certain parts corresponding to the touching moment are invariant across the demonstrations and other parts corresponding to the moment after retracting its arm are variant. The ability to extract this sort of structures specifying a task constraint from multiple demonstrations with trajectory-level uncertainty is important for intelligent robots to achieve adaptive behavior depending on the situations.

Although it is difficult to identify possible sources of fluctuations and the true characteristic of such dynamically changing stochastic structures of the fluctuations unlike the numerical experiments, the S-CTRNN demonstrates its ability to extract and reproduce the structures via the predictive learning with uncertainty estimation. The results also show that such structures specifying the task constraint of reaching movement can be represented not only in learned situations but also in unlearned situations by utilizing generalization abilities of the trained network.

4.2 Methods

4.2.1 Design of Reaching Experiment

A small humanoid robot "NAO" developed by *Aldebaran Robotics* was employed for the experiment. The task for the robot was to learn reaching movement toward a target object affixed to a workbench (Fig. 5.2) by integrating visuo-proprioceptive states and contextual dynamics. The S-CTRNN was utilized as a generative model for learning to predict visuo-proprioceptive states.

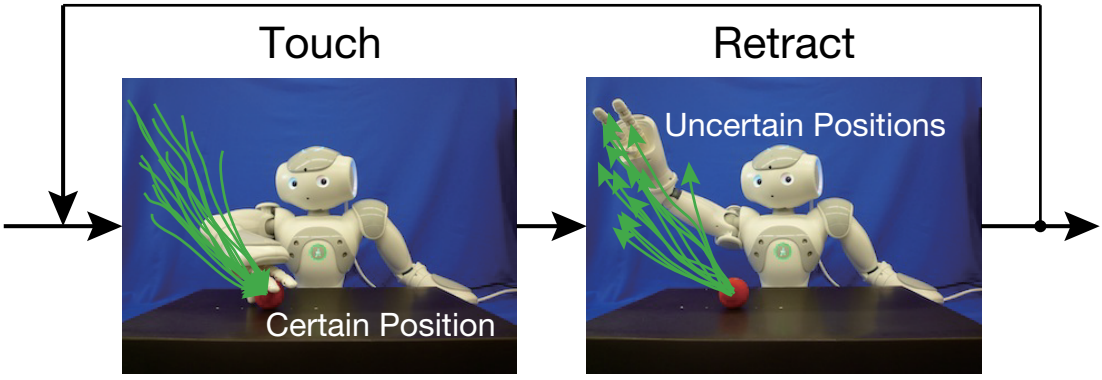


Figure 4.1 Movement sequence recorded during a tutoring session. The task for the robot was simply to touch a red sphere placed on a workbench with its right hand and then retract the hand. This touch-and-retract movement was repeated several times via kinesthetic teaching to record a target visuo-proprioceptive sequence that included various trajectories with a stochastic structure.

Multiple demonstrations of the reaching movement were performed by means of kinesthetic teaching, where the robot's arm was directly guided by a human tutor. During the demonstrations, the tutor was instructed to guide the robot's

hand precisely until it touched the target object and to subsequently retract the hand without following a precise trajectory. Therefore, the resultant hand movement represented a specific stochastic structure as illustrated in Fig. 5.2. The stochastic structure contains the certain hand position at the touching moment and uncertain positions after retracting the arm. It was expected that this sort of structure representing both the important and less important parts for specifying the task constraint of the reaching movement can be extracted via the predictive learning with uncertainty estimation by using the S-CTRNN. After the successful predictive learning, the robot equipped with the trained network was expected to be able to reproduce the tutored reaching movement together with the extracted structure.

The object was located at one of three positions (denoted by Positions 1–3) in the tutoring phase (Fig. 4.2), and it was affixed at the same position while the tutor repeatedly demonstrated the touch-and-retract movement. Here, Position 3 was the center of the workbench on the line of symmetry between the left and right sides of the robot.

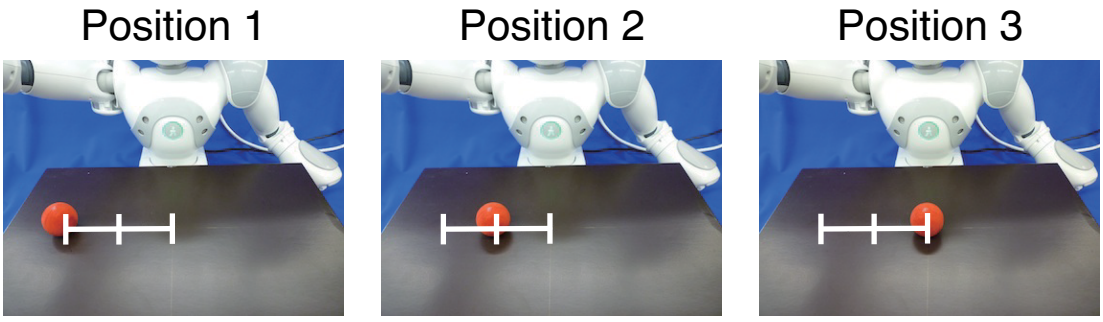


Figure 4.2 Three positions of the target object. The distance between two neighboring positions was 7 cm.

4.2.2 Experimental Procedure

The robotics learning experiment consisted of (1) recording visuo-proprioceptive sequences from the robot, (2) predictive learning with the S-CTRNN in an offline manner using the recorded target sequences, and (3) testing action generation in which the robot was controlled by the trained network.

Recording Visuo-Proprioceptive Sequences

In the first phase of data recording, the robot was directly tutored on its own movements in terms of the joint angles of the head and right arm for generating reaching movement. The touch-and-retract action guided by the tutor was repeated 15 times for one target sequence. Note that the touch-and-retract action in each target sequence was not always completely periodic, although the tutor attempted to generate precise periodic patterns by using a metronome. The actual learning took place with such training data with uncertainty. To ensure robust learning, 10 target sequences were recorded for each of the three object positions. Therefore, the total number of target sequences used in the learning phase was 30, for a total of 450 touch-and-retract actions. Each target sequence consisted of a time series of a two-dimensional vector representing head joint angles (yaw and pitch), and a four-dimensional vector representing right arm joint angles (shoulder pitch, shoulder roll, elbow yaw, and elbow roll) of the robot. The remaining joint angles were fixed.

The head joint angles were controlled to fixate automatically on the center of the target object, regardless of the robot's action both in the data recording phase with kinesthetic teaching and in testing phase with the trained network. Therefore, the head direction provided visual information about the relative position of the object in this experiment. The target visuo-proprioceptive sequences were recorded every half a second, and the length of each sequence $T^{(s)}$ was about 180.

Predictive Learning

In the second phase of the predictive learning, visuo-proprioceptive states recorded in target sequences were mapped to values ranging between -0.8 and 0.8 because the activation function of the output units of the S-CTRNN was tanh (ranging between -1.0 and 1.0). The S-CTRNN was trained to predict the mean and variance states of the visuo-proprioceptive state at the next time step by integrating the current state in the target sequence and contextual dynamics stored in the network.

Testing Action Generation

In the third phase of actual action generation, the robot was tested for reproduction of the learned reaching movement with the object located in learned as well

as in new positions. Changes in the object position represented by the head joint angles were sent back to the network as visual sensory feedback. As mentioned above, in both the first recording and third testing phases, the head joint angles were controlled to automatically fix on the center of the target object, regardless of the robot's movement. For action generation, initial internal states $u_{0,i}^{(s)}$ of context units corresponding to each sequence s were not used, and instead the mean value $\hat{u}_i$ of the initial states of all target sequences was used. Therefore, differences in behavior trajectories are not due to the sensitivity to initial conditions of the trained network but arise from differences in the open-loop visual states (i.e., object position) and the Gaussian noise added to the closed-loop proprioceptive states.

4.2.3 System Architecture

Figure 4.3 presents an overview of the system architecture for the action generation after the predictive learning.

Inputs to the S-CTRNN were in the form of visual states $\boldsymbol{x}_t^{(\mathrm{v})}$ and proprioceptive states $\boldsymbol{x}_t^{(\mathrm{p})}$. It should be noted that the superscripts here such as $(\cdot)^{(\mathrm{v})}$ and $(\cdot)^{(\mathrm{p})}$ are used for discriminating the modalities and the index for specifying the sth sequence is not written for simplicity. The network predicted the input states for the next time step as output states $\boldsymbol{y}_t^{(\mathrm{v})}$ and $\boldsymbol{y}_t^{(\mathrm{p})}$, and also predicted the variances $\boldsymbol{v}_t^{(\mathrm{v})}$ and $\boldsymbol{v}_t^{(\mathrm{p})}$ corresponding to each output. The predicted proprioceptive states $\boldsymbol{y}_t^{(\mathrm{p})}$ with the addition of Gaussian noise $\boldsymbol{\epsilon}_t^{(\mathrm{p})}$ with the predicted variance $\boldsymbol{v}_t^{(\mathrm{p})}$ was fed into the next input state, and these states 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 states $\boldsymbol{y}_t^{(\mathrm{v})}$ and variance $\boldsymbol{v}_t^{(\mathrm{v})}$ 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:

$$\begin{cases} \boldsymbol{x}_t^{(\mathrm{p})} = \boldsymbol{y}_{t-1}^{(\mathrm{p})} + \boldsymbol{\epsilon}_{t-1}^{(\mathrm{p})}, \\ \boldsymbol{x}_t^{(\mathrm{v})} = \hat{\boldsymbol{y}}_{t-1}^{(\mathrm{v})}, \end{cases} \tag{4.1}$$

where $\hat{\boldsymbol{y}}_{t-1}^{(\mathrm{v})}$ is an actual visual feedback at time step t that corresponds to the target state at the previous time step $t-1$.

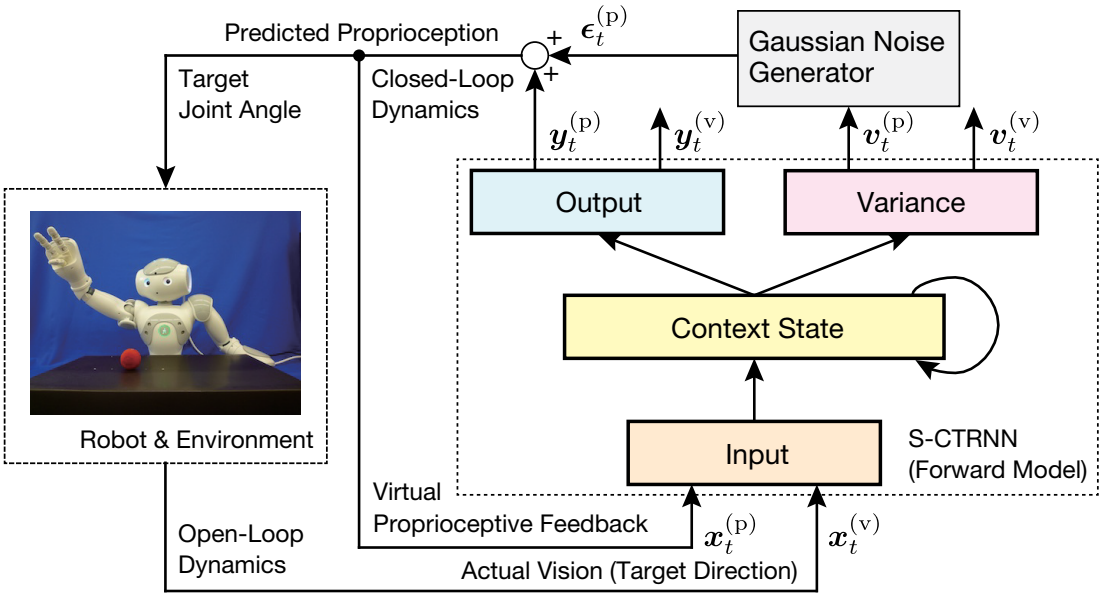


Figure 4.3 System architecture for action generation in reaching task. The S-CTRNN model was used as a forward model for controlling the robot. Inputs to the network were in the form of visuo-proprioceptive states $\boldsymbol{x}_t^{(\mathrm{v})}$ and $\boldsymbol{x}_t^{(\mathrm{p})}$. The network was trained to predict the inputs for the next time step. The predicted proprioceptive state $\boldsymbol{y}_t^{(\mathrm{p})}$ with addition of Gaussian noise $\boldsymbol{\epsilon}_t^{(\mathrm{p})}$ with the predicted variances $\boldsymbol{v}_t^{(\mathrm{p})}$ was sent to the robot in the form of target joint angles, which acted as motor commands for the robot to generate movements. The addition of Gaussian noise enables the network to generate output states with fluctuations whose stochastic structure is similar to that in the training data.

4.2.4 Parameter Settings for Predictive Learning

The number of input, output, and variance units were $N_{\mathrm{I}} = N_{\mathrm{O}} = N_{\mathrm{V}} = 6$, respectively. These were determined by the dimensionality of the target sequences containing the time series of the two-dimensional visual states and four-dimensional proprioceptive states. The delay length that determines the relationship between input states and target states was set to $\xi = 1$, namely, $\boldsymbol{x}_t^{(s)} = \hat{\boldsymbol{y}}_{t-1}^{(s)}$. The number of context units, the time constant of the context units, and the variance of the initial internal states of the context units were chosen to be $N_{\mathrm{C}} = 30$, $\tau_i = 2$ ($i \in I_{\mathrm{C}} = \{1, 2, \cdots, N_{\mathrm{C}}\}$), and $\sigma_{\mathrm{IS}}^2 = 0.001$, respectively. The parameter for determining the learning rate and the momentum term were chosen to be $\tilde{\alpha} = 0.0001$ and $\eta = 0.9$, respectively.

The network parameters optimized in the learning process including synaptic weights, biases, initial internal states of context units were initialized in the way written in Chapter 2 (Section 2.7).

4.3 Results

The S-CTRNN was trained for 500,000 training epochs with the open-loop mode (2.25a). After the training, the robot equipped with the trained network was evaluated whether it was able to reproduce the learned movement together with the stochastic structures latent in the visuo-proprioceptive sequences. Adaptation to unlearned situations was also evaluated.

4.3.1 Extraction of Task Constraints

Figure 4.4 illustrates an example of visuo-proprioceptive sequences in the training data, output states and variance predicted by the trained S-CTRNN. As can be seen from the figure, the output states of the trained network with closed-loop dynamics with addition of Gaussian noise with the predicted variance reproduce the stochastic structure of the training data which contains cyclic patterns consisting of invariant and variant parts. 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). In other words, the task constraints of reaching movement

latent in multiple demonstrations was able to be extracted by via the predictive learning with uncertainty estimation.

Figure 4.5 shows snapshots capturing the moments of touching and after retracting for each object position. As seen in the figure, the position of the robot's hand is converged to the point corresponding to the position of the object. In contrast, it diverges when the robot retracts its hand after touching the object.

4.3.2 Generalization and Adaptation Abilities

The robot equipped with the trained S-CTRNN was able to produce the reaching movement 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 position 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 adaptively touch the object even if the object was shifted from its original position when the robot started to move.

Figure 4.6 presents phase plots of one of training data, the output and two selected context states of the trained network for each position, including cases where the object is located between learned positions. In comparing Fig. 4.6, the phase plots of the training data and those of the output of the trained network share similar dynamic stochastic 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 structures can be seen in the 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 structures.

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-varying means and variance states. Furthermore, the

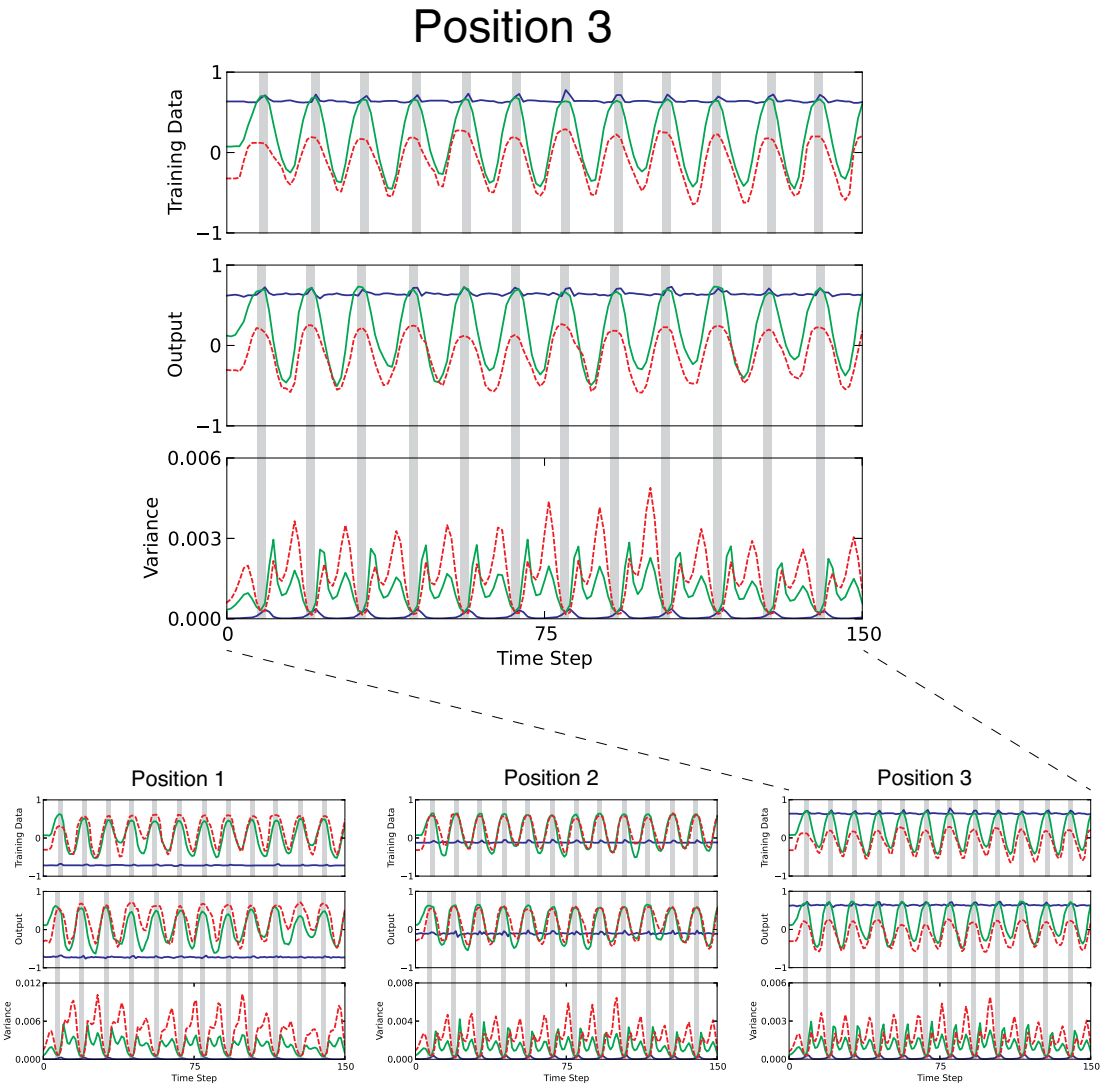


Figure 4.4 Temporal sequences obtained in the experiment for each object position (Positions 1–3). Training data (first row), output of the trained S-CTRNN with closed-loop dynamics with addition of Gaussian noise with the predicted variance (second row), and predicted variance of the trained network (third row). Each panel shows a plot of one out of two visual dimensions (blue: head pitch) and two out of four proprioceptive 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.

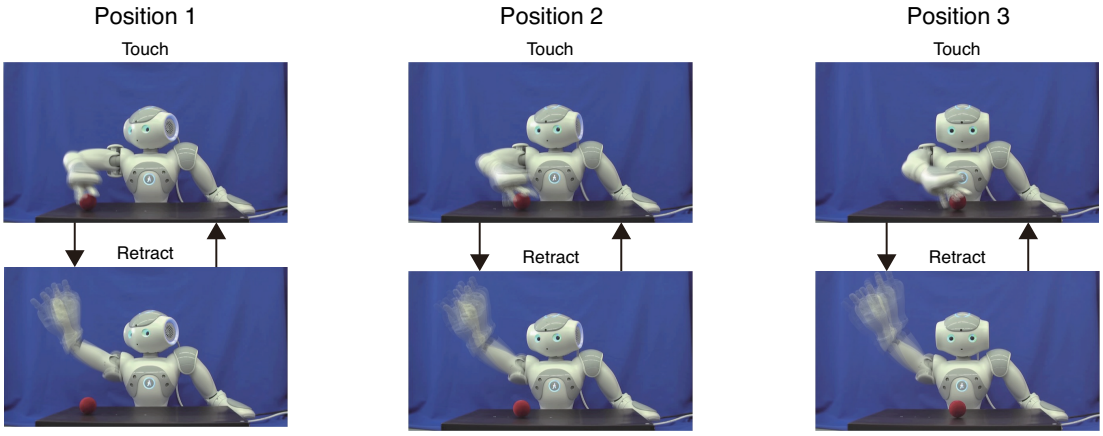


Figure 4.5 Snapshots of action sequences performed by NAO controlled by the trained S-CTRNN for each object position (Positions 1–3). The upper panels correspond to movement directed towards touching the object, and the lower panels correspond to the moment after the robot retracts its arm.

generalization and adaptation capabilities via learning for new sensory inputs (e.g., new object positions) were also confirmed.

4.4 Discussion and Conclusions

In the experiment, S-CTRNN was able to reproduce fluctuating behavioral sequences with trajectory-level uncertainty demonstrated by a human tutor. The robot controlled by the trained network reproduced the reaching movement and exhibited fluctuations with a structure similar to that of the demonstrated 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 demonstrated ones. Such behavioral trajectories were generated successfully even for new sensory inputs through generalization of learning. Moreover, the robot was able to adapt to perturbations such as sudden changes in the object position during action generation. Namely, the results demonstrated the extraction of trajectory-level uncertainty via predictive learning and adaptive reproduction of learned skills (or action primitives) in unlearned situations. These results suggest that the proposed model can achieve generalization and adaptation to some extent by learning.

Several issues remain to be examined in future studies. One such issue is

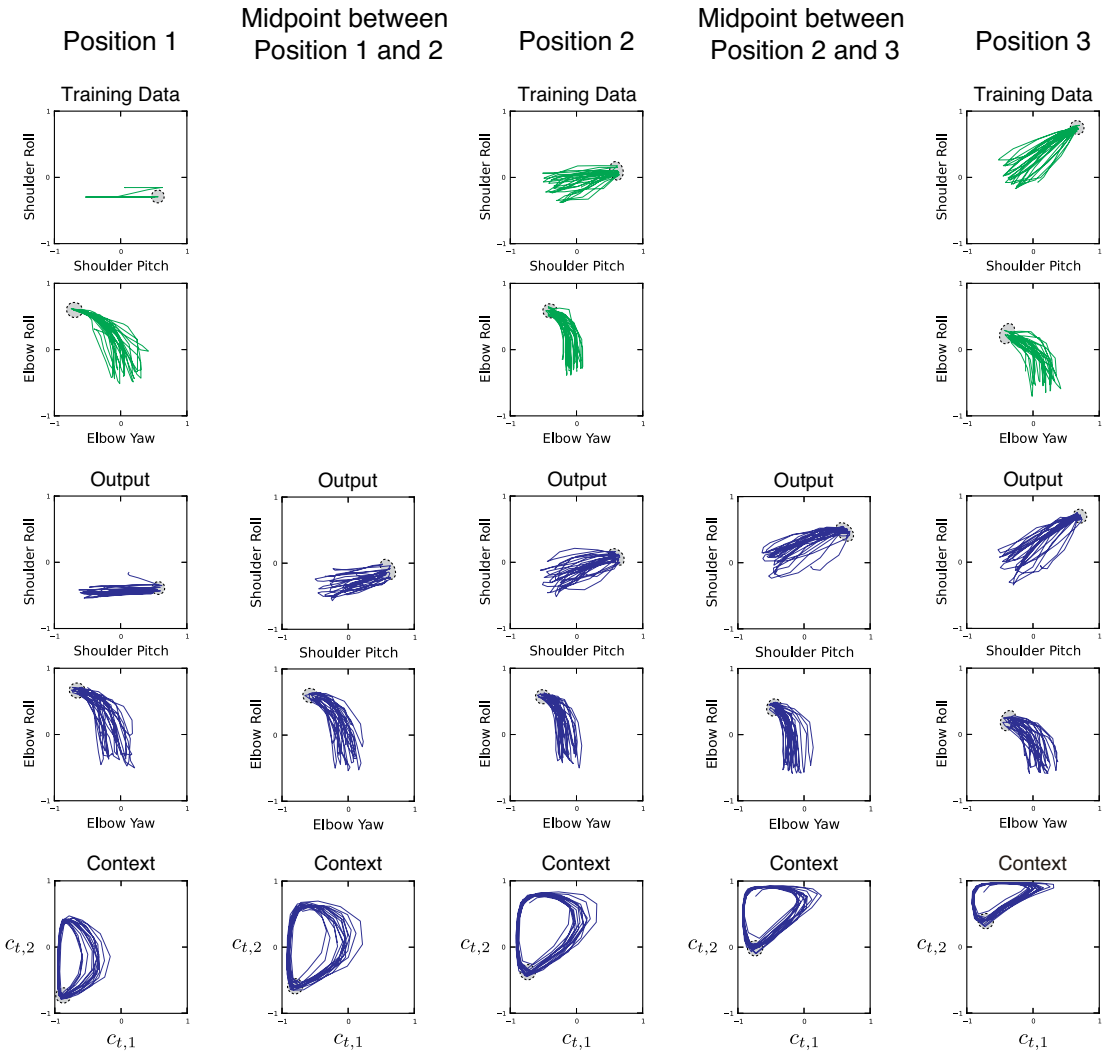


Figure 4.6 Phase plots of the training data, the output and the two selected context states of the trained network for each object position. Shoulder space of the training data (first row), elbow space of the training data (second row), shoulder space of the output (third row), elbow space of the output (fourth row), and context space (fifth row). The gray areas correspond to the moments at which the robot touched the object.

the *temporal registration* problem. In the probabilistic approach employing a combination of GMM and GMR [30,93,94], 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. [95] 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 target 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 [80, 96], 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 [94], the learned variance information was interpreted as a measure of the importance of parts of the pouring movement. In parts with large 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 this chapter, it is considered that the 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.

Chapter 5

Predictive Learning to Develop Flexible Behavior

5.1 Introduction

In the preceding chapter, the S-CTRNN was applied to robot learning in the context of learning to develop adaptive behavior. The generalization and adaptation abilities of the network via the predictive learning with uncertainty estimation was demonstrated. This chapter focuses on the development of reactive and proactive behavior. These different behavior generation schemes enable cognitive agents (or intelligent robots) to realize flexible behavior. More precisely, we consider how adequate actions are selected and generated in a flexible manner based on acquired generative models via the predictive learning with uncertainty estimation.

The S-MTRNN, which is an extension of the S-CTRNN used in Chapters 3 and 4, is employed as a generative model to achieve hierarchical predictive representation about complex visuo-proprioceptive states consisting of the sequential combination of some primitive patterns. The higher-level network with slow dynamics represents sequential information with abstraction of primitives and the lower-level network with fast dynamics represents specific profiles of the primitives. Furthermore, we consider not only optimizing parameters for learning, but also inferring internal states by the ERS for dynamic recognition of situational changes.

In the experiment, the same humanoid robot as the one used in Chapter 4 is used for a cooperative interaction task. The robot equipped with the S-MTRNN is required to learn to interact with the other pre-programmed robot.

A computer program probabilistically determines the other's action primitive from repertoire of primitives. Because the action primitives represented by behavioral (visuo-proprioceptive) trajectories are fixed, there is no trajectory-level uncertainty. However, the transitions of these events (or fixed primitives) are uncertain. Therefore, the robot controlled by the S-MTRNN is necessary to deal with event-level uncertainty about the other's behavior represented as visual states. The robot interacting with the other robot is expected to flexibly change action primitives in a reactive manner based on the current sensory states when external situations are estimated as uncertain. In contrast, it is expected to change the primitives in a proactive manner based on the own prior intentional states and predictions when the situations are estimated as certain.

Experimental results demonstrate that these differences can be emerged in a self-organized fashion in accordance with learning conditions. Specifically, a probabilistic generative model for reactive behavior is developed when the sensitivity to initial conditions of context units is allowed during the learning process. On the other hand, a deterministic generative model for proactive behavior is developed when the sensitivity is not allowed. Based on an analysis of the experimental result for the different learning conditions, this chapter discusses the mechanism of learning to develop reactive and proactive behavior for realizing flexibility.

5.2 Methods

5.2.1 Design of Interaction Experiment

Two humanoid robots "NAO", which have already been introduced in the preceding chapter, were employed for the experiment (Fig. 5.1). One robot called the "self-robot" was required to generate action sequences flexibly corresponding to those generated by the "other-robot" via learning in a supervised manner. More specifically, the self-robot faced the other-robot and learned to predict how the positions of an object held by the other-robot change in time based on visual sequences. It also learned its own corresponding arm movements in terms of proprioceptive sequences.

Figure 5.2 (left) shows a schematic illustration of the task. The self-robot was controlled by the S-MTRNN model and the other-robot followed action sequences pre-programmed by the experimenter; for simplicity, only the self-robot was re-

Figure 5.1 Experimental environment for interaction task. Two NAO robots were utilized for the experiment.

quired to generate flexible behavior and the other-robot's action sequences were not affected by the self-robot. In the task, the other-robot arbitrarily repeated action primitives involving moving a colored object either to the left (labeled as "L") or to the right (labeled as "R") from the view point of the self-robot. The self-robot was required to generate corresponding behavior in terms of moving its right arm in the same direction and simultaneously with the other-robot at each time step. For the purpose of simultaneous action generation, the self-robot was required to predict the direction in which the object was about to be moved before the other-robot actually generated its movement.

The self-robot acquired this skill in a supervised learning phase. In the context of adaptation to the other-robot's behavior, the task for the self-robot can be considered as a cooperative interaction task in which the self-robot attempted to generate cooperative behavior with the other-robot. It should be noted that, after the learning phase, the S-MTRNN model implemented in the self-robot cannot predict the visuo-proprioceptive sequence completely in the test phase in this task because the decision about whether to move the object to the left or to the right at each branching level is generated at random by the pre-programmed other-robot. In this context, the objective of the experiment was to examine how, after learning, the self-robot can generate cooperative behavior in a flexible manner even when the self-robot occasionally fails to make a correct prediction about the branching points.

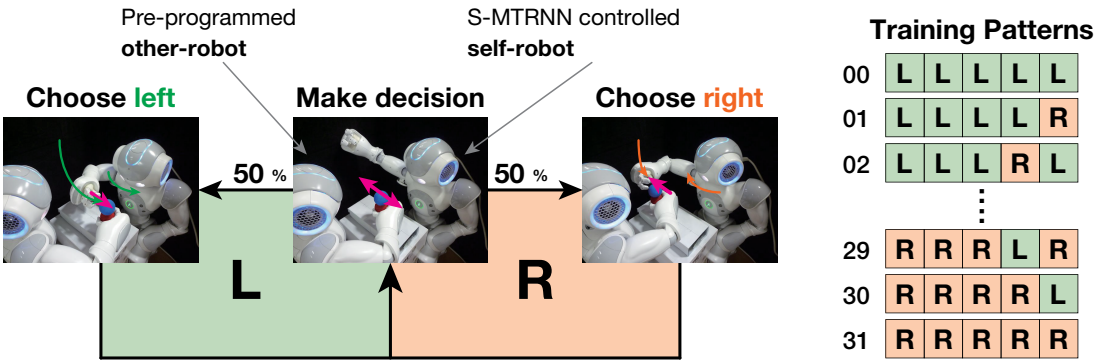


Figure 5.2 Task design. The left part of the figure shows a schematic of a cooperative interaction task. The "self-robot" controlled by the S-MTRNN and a pre-programmed "other-robot" were used in the experiment. In the task, the other-robot arbitrarily moved a colored object either to the left or to the right with equal probability ($P = 0.5$). The self-robot was required to learn to flexibly interact with the other-robot in terms of moving its right arm in the same direction at the same time as the other-robot (left and right photographs in the figure). The right part of the figure illustrates the 32 training patterns used in the experiment. These patterns covered all the transition patterns in the case where the transition was repeated five times in each sequence.

5.2.2 Experimental Procedure

The robotics learning experiments consisted of (1) recording visuo-proprioceptive training sequences from the self-robot interacting with the other-robot, (2) predictive learning with the S-MTRNN in an offline manner using the recorded target sequences, and (3) testing action generation in which the self-robot was controlled by the trained network.

Recording Visuo-Proprioceptive Sequences

In the first phase of data recording, the self-robot was directly tutored on its own movements in terms of head angle and right arm posture for generating cooperative behavior that matched the actions of the other-robot for action primitive sequences consisting of five-level decision branching. Each branch corresponded to moving the object to the left or to the right. In the current experiment, 32 pattern sequences covering all possible five-level branching sequences (such as "RRLLR") were recorded as a training data set (see the right hand part of Fig. 5.2). Each sequence consisted of a temporal sequence of a two-dimensional vector representing the object's center position in visual images obtained from the

camera mounted on the tutored self-robot, a two-dimensional vector representing head joint angles (yaw and pitch), and a four-dimensional vector representing right arm joint angles (shoulder pitch, shoulder roll, elbow yaw, and elbow roll) of the self-robot. These data were recorded 10 times a second, and the length of each sequence $T^{(s)}$ was about 470.

Predictive Learning

In the second phase of the network training, visuo-proprioceptive states recorded in training sequences were mapped to values ranging between -0.8 and 0.8 because the activation function of the output units of the S-MTRNN was tanh (ranging between -1.0 and 1.0).

Two different values for the variance of the initial internal states of the SC units were employed: one with a small variance ($\sigma_{\mathrm{IS}}^2 = 0.00001$) and one with a large variance ($\sigma_{\mathrm{IS}}^2 = 10$). For simplicity, we refer to the former and the latter cases as the "narrow IS distribution" and the "wide IS distribution", respectively, where IS means initial state. In the narrow IS distribution, the initial sensitivity of the network explained in Chapter 2 (Section 2.2.3) seems to be utilized less frequently to differentiate between the initial states for generating different sequences, and in the wide IS distribution, the sensitivity seems to be utilized more frequently (see Fig. 5.3). By imposing these different learning conditions, the effect of the difference in the IS distribution on the development of different behavior generation schemes was investigated. The S-MTRNN was trained to predict the mean and variance states of the visuo-proprioceptive state at the next time step by using the current state in the training sequence.

Testing Action Generation

In the third phase of actual action generation, an arbitrary set of initial internal states of the SC units was given to the trained network. Changes in the environment, including changes in the object position and changes in the actual joint angle positions were sent back to the network as sensory feedback. In both the first and third phases, the self-robot's head joint angles were controlled to automatically fix on the center of the target object, regardless of the robot's behavior.

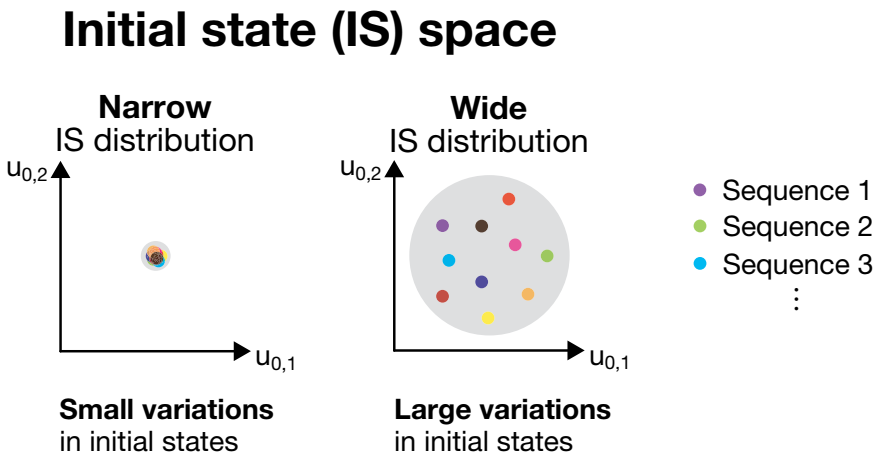


Figure 5.3 Initial state (IS) space of the higher-level network containing SC units. These initial conditions were optimized during learning and confer context sensitivity on the subsequent dynamics. Although the actual dimensionality of the space was 10 (the number of SC units), this space is shown here as a two-dimensional space for visualization purposes. Left: narrow IS distribution case, in which a small variance ($\sigma^2_{\mathrm{IS}} = 0.00001$) was employed; right: wide IS distribution case in which a large variance ($\sigma^2_{\mathrm{IS}} = 10$) was employed. This difference in variance affected the initial sensitivity of the S-MTRNN. By the "initial internal state" we mean the initial values of the internal states of the SC units ($u^{(s)}_{0,i}$). Each colored circle represents an initial internal state associated with a particular visuo-proprioceptive sequence.

5.2.3 System Architecture

Figure 5.4 presents an overview of the system architecture for the action generation after the predictive learning.

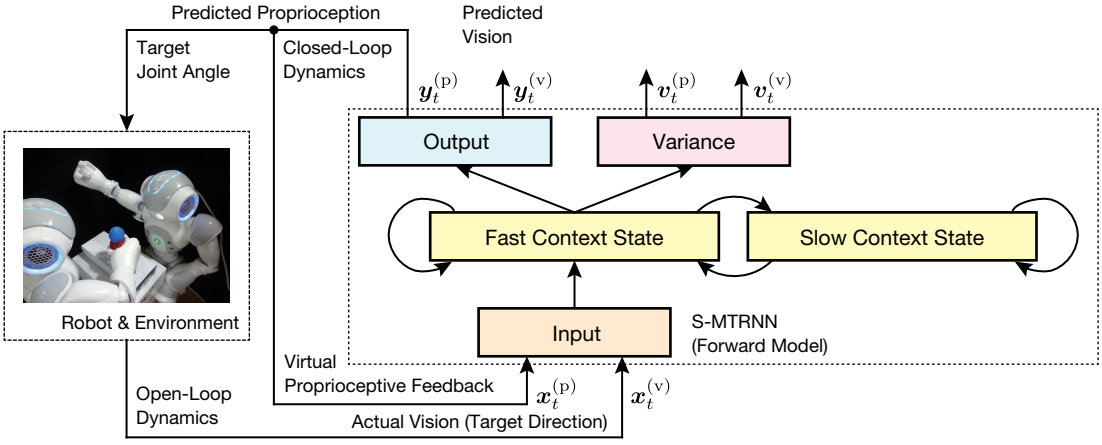


Figure 5.4 System architecture for action generation in interactive task. The S-MTRNN model was used as a forward model for controlling the self-robot. Inputs to the network were in the form of visuo-proprioceptive states $\boldsymbol{x}_t^{(\mathrm{v})}$ and $\boldsymbol{x}_t^{(\mathrm{p})}$. The network was trained to predict the inputs for the next time step. The predicted proprioceptive state $\boldsymbol{y}_t^{(\mathrm{p})}$ was sent to the robot in the form of target joint angles, which acted as motor commands for the robot to generate movements.

Inputs to the S-MTRNN were in the form of visual states $\boldsymbol{x}_t^{(\mathrm{v})}$ and proprioceptive states $\boldsymbol{x}_t^{(\mathrm{p})}$. It should be noted that the superscripts here such as $(\cdot)^{(\mathrm{v})}$ and $(\cdot)^{(\mathrm{p})}$ are used for discriminating the modalities and the index for specifying the sth sequence is not written for simplicity. The network predicted the input states for the time step $t+\xi$ as output states $\boldsymbol{y}_t^{(\mathrm{v})}$ and $\boldsymbol{y}_t^{(\mathrm{p})}$, and also predicted the variances $\boldsymbol{v}_t^{(\mathrm{v})}$ and $\boldsymbol{v}_t^{(\mathrm{p})}$ corresponding to each output. The previously predicted proprioceptive state $\boldsymbol{y}_{t-\xi+1}^{(\mathrm{p})}$ was fed into the next input state, and these states 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 states $\boldsymbol{y}_{t-\xi+1}^{(\mathrm{v})}$ and variances ($\boldsymbol{v}_{t-\xi+1}^{(\mathrm{v})}$ and $\boldsymbol{v}_{t-\xi+1}^{(\mathrm{p})}$) were not used for action generation because the joint angles of the robot's head were controlled independently, as mentioned above.

5.2.4 Parameter Settings for Predictive Learning

The number of the input, output, and variance units were $N_{\mathrm{I}} = N_{\mathrm{O}} = N_{\mathrm{V}} = 8$, respectively. These were determined by the dimensionality of the visuo-proprioceptive state of a humanoid robot (two-dimensional visual inputs, two-dimensional head joint angles, and four-dimensional right arm joint angles as described in the following section). The delay length that determines the relationship between input states and target states was set to $\xi = 5$, namely, $\boldsymbol{x}_t^{(s)} = \hat{\boldsymbol{y}}_{t-5}^{(s)}$. The numbers of FC and SC units were $N_{\mathrm{FC}} = 30$ and $N_{\mathrm{SC}} = 10$, respectively. The time constants of the FC and SC units were $\tau_{\mathrm{FC}} = 5$ and $\tau_{\mathrm{SC}} = 100$, respectively. The same parameters were employed for the training in both the narrow and wide IS distribution conditions for comparison.

In all experiments presented in this chapter, the parameter for determining the learning rate $\tilde{\alpha}$ and the momentum term were chosen to be $\tilde{\alpha} = 0.0001$ and $\eta = 0.9$, respectively.

The network parameters optimized in the learning process including synaptic weights, biases, and initial internal states of context units were initialized in the way written in Chapter 2 (Section 2.7). To accelerate network training, the adaptive learning rate scheme based on the works of Namikawa et al. [44, 71] was employed. Details are provided in Appendix D (Section D.1).

5.3 Results

The S-MTRNN with two different learning conditions, one with the narrow IS distribution ($\sigma_{\mathrm{IS}}^2 = 0.00001$) and the other with the wide IS distribution ($\sigma_{\mathrm{IS}}^2 = 10$), was trained for 500,000 training epochs with the open-loop mode (2.25a).

5.3.1 Reproduction of Visuo-Proprioceptive Sequences with Different Representations of Uncertainty

After the network training, the visuo-proprioceptive sequences produced by each trained network with a closed-loop generation were first analyzed before the self-robot commenced the actual cooperative interaction with the other-robot. In this generation mode, an optimized initial internal state of the SC units is set for the higher-level network, and the input state at the current time step is derived from the predicted mean value to which Gaussian noise with the variance predicted at the previous time step is added by using (2.25b). Because this generation

mode does not receive actual sensory feedback, the process can be regarded as a simulation of action generation or motor imagery [44, 74, 97] taking fluctuations into account.

The trained network can deterministically regenerate learned sequences if adequate initial states are acquired for each sequence [44] and if the estimated variances remain sufficiently small (almost zero) through time. If the estimated variances are non-zero, for example at decision branching points, the sequences are generated stochastically by the effect of Gaussian noise added in accordance with the predicted variance.

Figure 5.5A and 5.5B show examples of sequences reproduced by the trained network in the narrow and wide IS distribution conditions. The sequences include temporal sequences of sensory targets (training data), sensory (mean) predictions, variance predictions, SC states, and FC states obtained from the S-MTRNN with closed-loop generation in which the initial state of the higher-level network was set to the "RRLLR" sequence.

In the narrow IS distribution condition, the network was unable to reproduce any learned sequence associated with a particular initial state with closed-loop generation. In Fig. 5.5A, for example, although the initial state for the network was set with the values optimized for the "RRLLR" sequence in the learning process, the generated sequence was "LRLLR". The figure suggests that a large variance is predicted at each branching point, indicating that the network regards the forthcoming perceptual event as uncertain. This prediction of a large variance or small precision at each branching point is reasonable because the visual input derived from the other-robot's behavior is essentially unpredictable. It was confirmed that the network was able to flexibly produce various combinatorial sequences derived from visual perturbations caused by self-generated noise with the predicted variances added at each branching point, instead of utilizing sensitivity to initial conditions of the higher-level network. These results indicate that the network developed a probabilistic prediction model at the branching points and a deterministic one for the other segments. It can be said that the regenerated sequences contained deterministic chunks (moving to either the left or the right) with probabilistic transitions or high event-level uncertainty.

In contrast, in the wide IS distribution condition, the network was able to reproduce every learned sequence from the acquired initial state set in the higher-level network. In Fig. 5.5B it can be seen that the predicted variance at all times is nearly zero. These results indicate that the branching sequences are flexibly

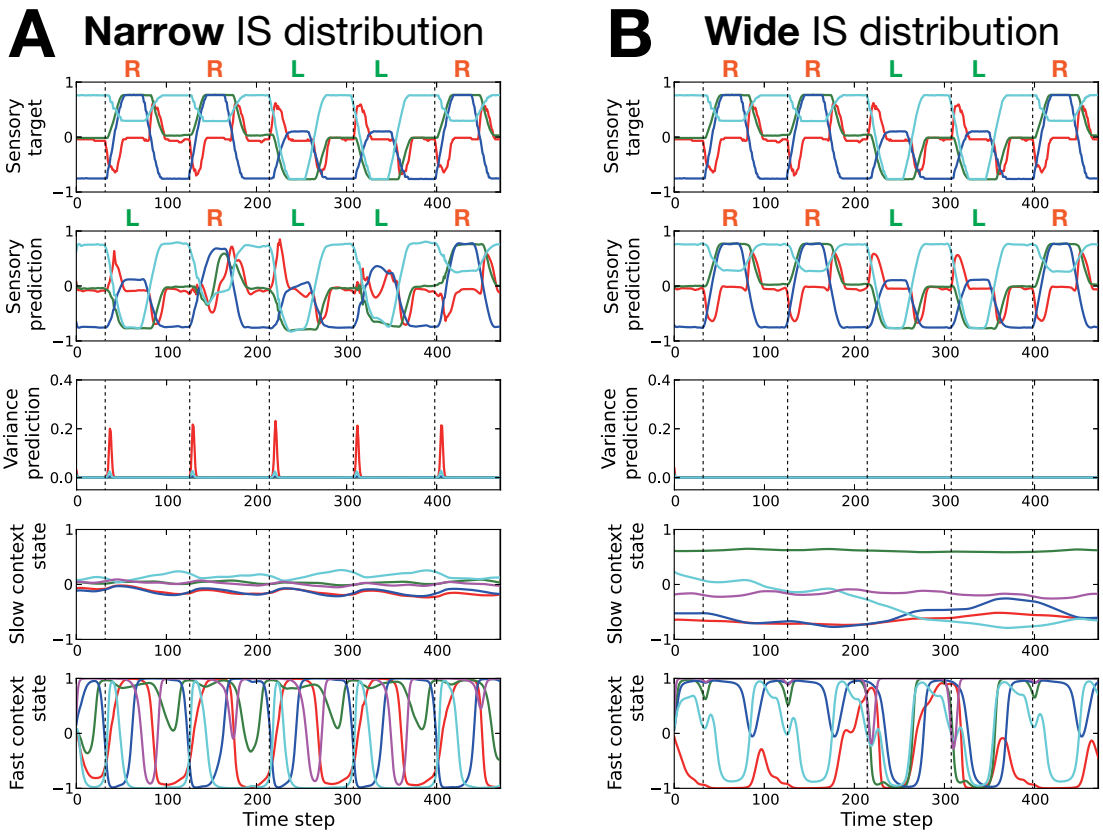


Figure 5.5 Temporal sequences obtained in the experiment. Temporal sequences of sensory targets (training data), sensory predictions (network mean output), variance predictions, SC states, and FC states during the closed-loop operation of the networks trained with (A) the narrow IS distribution and (B) the wide IS distribution. One time step corresponds to 100 ms. In the upper three panels, the red, green, blue, and cyan lines indicate the horizontal position of the object in a visual image, the head yaw angle, the shoulder pitch angle, and the elbow yaw angle, respectively. In the lower two panels, neural activities of five selected units (from 10 SC units and 30 FC units) are shown. The vertical dashed lines indicate branching points from which the object was moved either to the left or to the right by the other-robot in the training sequence. The labels over the panels of sensory targets and sensory predictions denote the action performed by the self-robot.

reproduced as deterministic dynamic sequences depending on the initial state.

The two IS distributions also show differences in terms of FC and SC unit states. In the narrow IS distribution condition, the initial value of each SC unit is not widely spread and is located near 0. The activities of both FC and SC units at the branching points seem to be almost the same, regardless of future sensory predictions. Therefore, it can be said that transitions are determined not by the internal context dynamics but by the self-generated noise, where the variance shows a sharp peak at the branching point. In contrast, in the wide IS distribution condition, both SC and FC units exhibit specific activation patterns. The dynamics of the SC units gradually change and those of the FC units have distinct forms at each branching point by which the subsequent sensory predictions can be discriminated. In summary, top-down prediction does not take place at branching points in the narrow IS distribution condition, whereas it does when using the deterministic neural dynamics developed with the initial precision characteristics in the wide IS distribution condition.

5.3.2 Action Generation Test

After evaluating the training results, experiments looking at actual cooperative interactions between the self-robot and the other-robot were performed. The self-robot was controlled by the system shown in Fig. 5.4. The other-robot was pre-programmed and its behavior was not affected by the self-robot.

To observe how the self-robot flexibly changes its behavior to interact with the other-robot's unpredictable behavior, an arbitrarily selected initial state for the trained S-MTRNN controlling the self-robot was set. Therefore, there is a discrepancy between the actions of the other-robot as anticipated by the self-robot and the actual actions generated by the other-robot during the interaction. Through the action generation test, it was investigated how unmatched internal context dynamics can be modified in order to adapt to the unpredictable behavior of the other-robot in the two learning conditions. Success or failure in this test phase was distinguished by means of the self-robot's hand position. A movement was judged to be successful when the hand was moved from the center of the body to the object's side, while it was judged to be a failure when the hand moved to the opposite side of the object or collided with the other-robot.

Figure 5.6A and 5.6B show examples of temporal sequences of online-sensory predictions, variance predictions, prediction errors, SC states, and FC states ob-

tained from the S-MTRNN for narrow and wide IS distribution conditions implemented on the self-robot. In this example, the initial state for the self-robot was set with the values optimized for the "LLRRL" sequence in the learning process, whereas the other-robot was programmed to generate the "RRLLR" sequence for both the narrow IS and the wide IS distribution conditions. This setting means that if a particular initial state encodes the learned action sequence of "LLRRL", this initial state can regenerate the same sequence for the self-robot's action plan which becomes exactly opposite to the action program to be generated by the other-robot.

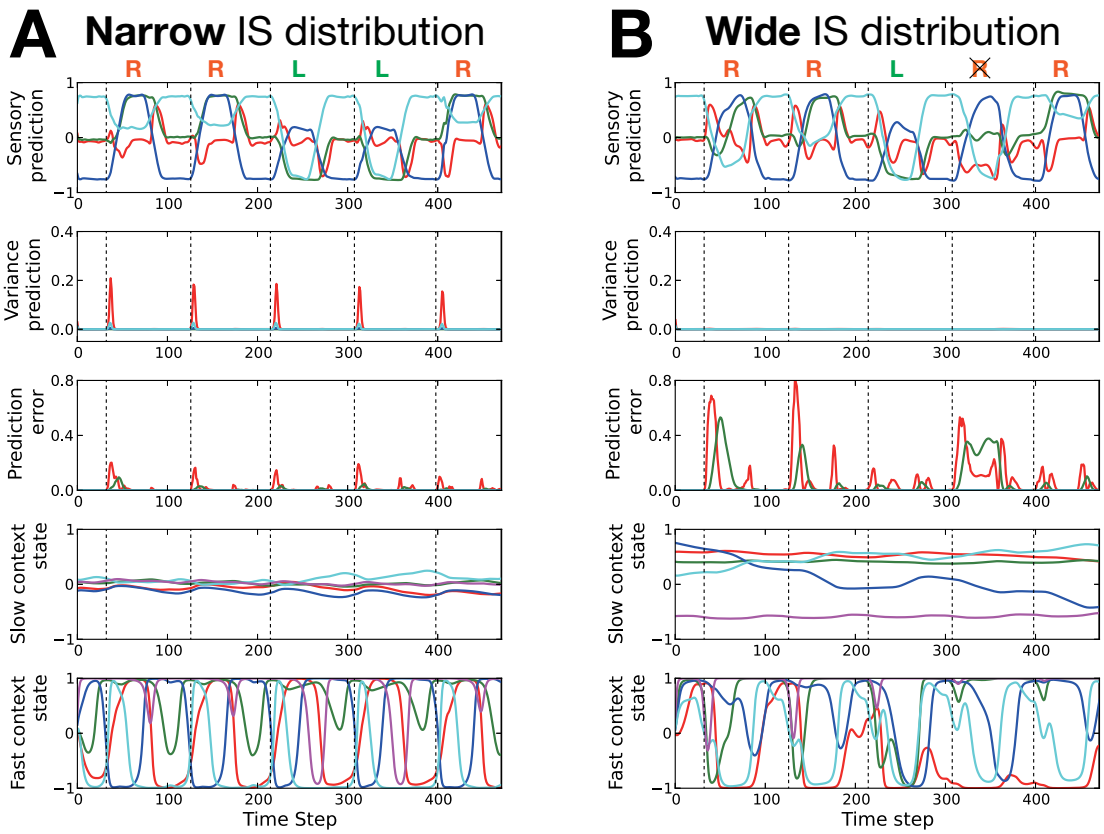


Figure 5.6 Temporal sequences obtained in the experiment. Temporal sequences of one-step sensory predictions, variance predictions, prediction errors, SC states, and FC states during action generation of the robot using the network trained with (B) the narrow IS distribution and (C) the wide IS distribution. The format of these panels is the same as that in Fig. 5.5B and 5.5C, except that here prediction errors are shown instead of sensory targets. In the case of the wide IS distribution, the cross on one label "R" represents a failure to predict the behavior of the other-robot, in which the hand of the self-robot collided with that of the other-robot.

In the narrow IS distribution condition, it was observed that the self-robot succeeded to generate the "RRLLR" sequence which corresponds to the sequence generated by the other-robot without any failure movements. This is evidenced by the observation that the one-step prediction profile is rather similar to the sensory target profile of the same sequence shown in Figure 5.5A. Also, it can be seen that some prediction errors were generated at each branching point. The states of both SC and FC units at each branching point are almost the same.

In contrast, for the wide IS distribution condition, the one-step prediction sequence was significantly poorer than that for the narrow IS distribution condition. In fact, the prediction error at each branch point often became significantly larger than the one in the narrow IS distribution condition. In this situation, the movement of the self-robot became inflexible. Although the self-robot seemed to try to follow the movements of the other-robot, its movements were significantly delayed. Furthermore, after three transitions between action primitives (i.e., after 300 time steps), the hand of the self-robot collided with that of the other-robot because the two robots moved their arms in opposite directions.

At first glance, it may appear counterintuitive that the narrow IS distribution can cope with violations of top-down predictions, whereas the wide IS distribution could not. Heuristically, one can understand this as follows: because we are optimizing the dynamics through the parameters, then the self-robot (after learning) is only optimal when the world behaves as expected. Crucially, these expectations include beliefs about precision. Therefore, paradoxically, an agent with a narrow IS distribution at the highest level learn that (precise) beliefs at lower levels can be violated and therefore contextualize sensory information by modulating the precision of prediction errors at those levels. In other words, only an agent with a narrow IS distribution can recognize its beliefs at lower levels are not always true.

5.3.3 Effect of Differences in Uncertainty Estimation on Reaction Times

To evaluate the effect of differences in uncertainty estimation on cooperative interactions, the reaction time of the self-robot, which is the number of time steps before the self-robot generated cooperative action primitives corresponding to the other-robot's action, was measured for the two learning conditions. In each condition, two initial types of state for the trained network were considered. In one case

the initial state corresponded to the other-robot's action sequence (this is referred to as the "corresponding IS" case) and in the other case an arbitrarily selected initial state was used (this is referred to as the "non-corresponding IS" case). In the current experiment, the initial state optimized for the "LLRRL" sequence was adopted for the non-corresponding IS case regardless of the other-robot's action sequences. This analysis was conducted not on physical interaction results but on ones simulated by using a set of recorded data and 10 trained sample networks with differently randomized initial parameters for each IS distribution condition. For details of the measurement, please refer to Appendix D (Section D.2). Mean reaction times over the 10 trained sample networks were computed, each of which generated 32 sequences including all combinations of the five transitions of the two action primitives (or 160 branches), for the corresponding and non-corresponding IS cases in each learning condition.

Figure 5.7 shows the computed mean reaction times in each case and results of t-test. In the narrow IS distribution condition estimating high event-level uncertainty, there is no significant difference ($t(18) = 0.71$, n.s.) in reaction times between the corresponding IS case ($M = 17.77$, $SD = 4.37$) and the non-corresponding IS case ($M = 17.80$, $SD = 4.36$). This result indicates that initial precision characteristics were no longer utilized on learning of different visuo-proprioceptive sequences and that the self-robot's behavior after learning relied not on internally generated context dynamics but on externally given sensory inputs. In contrast, in the wide IS distribution condition estimating low event-level uncertainty, there is a significant difference ($t(18) = 11.63$, $p < 0.001$) in reaction times between the corresponding IS case ($M = 0.004$, $SD = 0.008$) and the non-corresponding IS case ($M = 36.64$, $SD = 4.19$). In both the corresponding and non-corresponding IS cases, there are significant differences of reaction times between the narrow and wide IS distribution conditions ($t(18) = 12.21$, $p < 0.001$; $t(18) = 9.34$, $p < 0.001$; respectively). The shortest mean reaction time obtained in the wide IS distribution condition indicates that when the network was placed in the corresponding initial state, the action generation of the self-robot was almost synchronized with that of the other-robot. On the other hand, the longest mean reaction time obtained in the wide IS distribution condition indicates that a long time was required to revise behavioral contexts when the self-robot's anticipation derived from the initial state failed.

These differences observed between the two learning conditions can be attributed to the different neural dynamic structures developed for these condi-

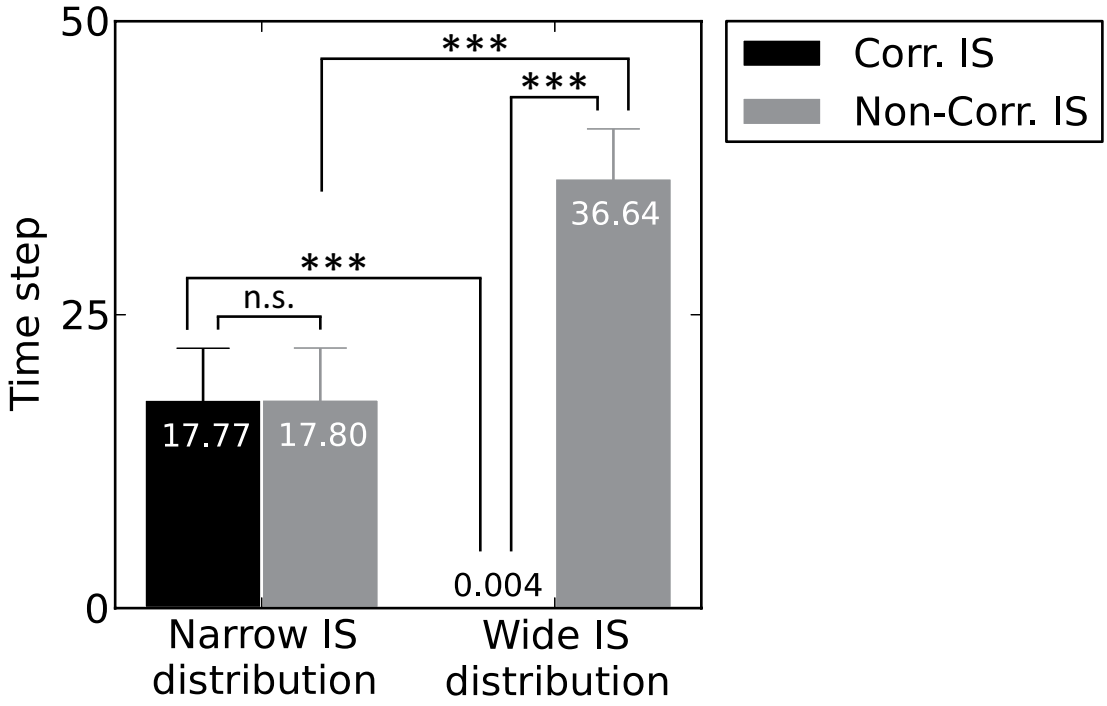


Figure 5.7 Reaction times of the self-robot during action generation. Times are given for the results of the simulated self-robot's action generation using the network trained with the narrow and wide IS distribution conditions. Bars and numbers in the graph correspond to mean reaction times over 10 trained sample networks for each IS distribution condition, each of which generated 32 sequences including the five transitions of the two action primitives. Black bars show the reaction times for initial states corresponding to the other-robot's action sequences and the gray bars show times for an arbitrarily selected (non-corresponding) initial state. In this case, the initial state optimized for the "LLRRL" sequence was adopted regardless of the other-robot's action sequences. Error bars indicate the standard deviation. Stars indicate a significant difference (***: $p < 0.001$) and "n.s." indicates no significant difference.

tions. In the case of the probabilistic dynamic structure with high event-level uncertainty developed in the narrow IS distribution condition, the behavior of moving either to the left or to the right is determined simply by following the other-robot by means of a sensory reflex without any top-down prior based on initial states. Therefore, the difference in initial states did not affect the self-robot's behavior or its reaction times. In contrast, in the case of the deterministic dynamic structure with low event-level uncertainty developed to be sensitive to the initial state, the top-down prior at the branching point is too strong to be modified by the sensory input. Therefore, when corresponding initial states were given to the network, it worked positively toward the realization of cooperative interactions without any time delay. However, when the non-corresponding initial states were given, the self-robot required a long time to adapt to the unanticipated actions of the other-robot and sometimes flexible modification was not achieved. To consider this problem, the effect of introducing an additional neural mechanism of bottom-up recognition by using error information was examined.

5.3.4 Action Generation with ERS for Proactive Behavior

As we learned from the experiment in the preceding subsection, when the top-down prior was too strong, a simple sensory reflex was insufficient for revising the internal neural dynamics. To solve this problem, the ERS or dynamic recognition method introduced in Chapter 2 (Section 2.10.2) was applied into the trained network and reconducted an action generation test in the wide IS distribution condition. Figure 5.8 shows a schematic illustration of the open-loop generation with ERS.

For updating internal states of the SC units using ERS, the likelihood L and the learning rate α in (2.19) are replaced with L_{reg} and α_{reg}, respectively. The learning rate was set to the same value used in the updating of the initial internal states of the SC units. During the error regression process, the adaptive learning rate scheme was not used for real-time computation and, therefore, the value was fixed.

Figure 5.9 shows an example of the temporal sequences of sensory predictions, variance predictions, prediction errors, SC states, and FC states obtained from the S-MTRNN with ERS with a wide IS distribution condition implemented on the self-robot. Clearly, the SC states in the gray areas change in a discontinuous manner. Modulating the higher-level SC states in this way by using ERS caused

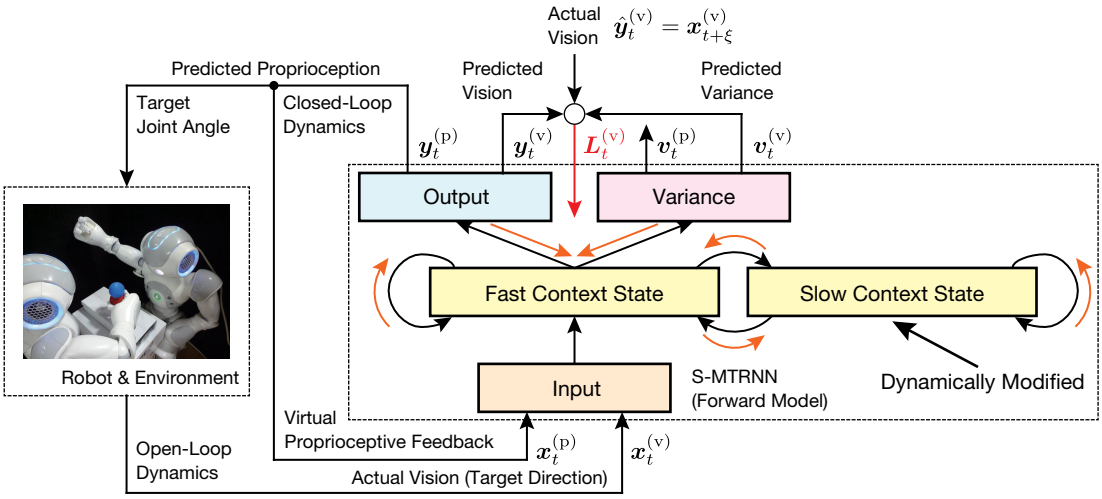


Figure 5.8 Open-loop generation with ERS. The internal states of the SC units are dynamically modified by using the gradient ascent method with BPTT (orange arrows) to maximize the likelihood (red arrow).

drastic changes in lower-level network activity, including the FC states and sensory predictions. Through these processes, the prediction errors were rapidly suppressed, and thus the self-robot was able to revise its behavioral context immediately after encountering unanticipated perceptual events.

To clarify the dynamic process of the regression by which the record of past states in the window can be overwritten and that of the prediction in which future plans can be modified by the regression, the states for time steps 175 to 265 were extracted from Fig. 5.9. Figure 5.10 shows three sets of the states when the current time step (the window head) is 221, 224, and 227, corresponding to the "pre-modification," "modification," and "post-modification" phases, respectively. In the figure, the dynamically sliding windows are shown as gray frames. The figure shows the states in the past (left side of the window head) with solid lines, in the future (right side of the window head) with dashed lines, and at the present (the window head) with labels "Now" and specific time indexes indicating the boundary between the past and future states. It should be noted that although the range of the time step of each panel is the same (from 175 to 265), not only future predicted states but also the record of the past states at the same time step can differ from each other because immediate past states within the time window can be overwritten by revising the internal neural dynamics using ERS. Past states outside of the window are not affected by the regression dynamics

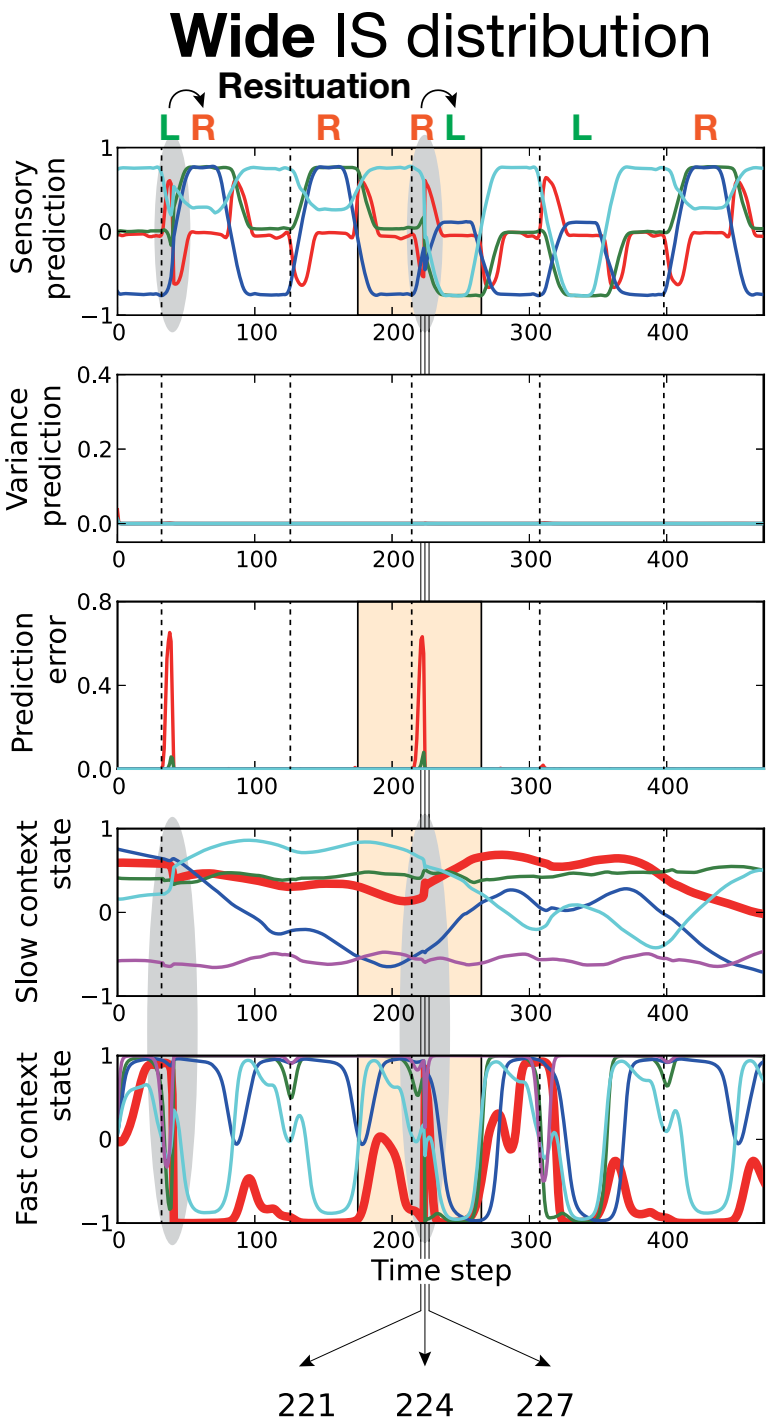


Figure 5.9 Temporal sequences obtained in the experiment. Temporal sequences of one-step sensory predictions, variance predictions, prediction errors, SC states, and FC states during action generation with error regression for a robot using the network trained with the wide IS distribution. The format of these panels is the same as that in Fig. 5.6A and 5.6B. Extracted states for time steps 175 to 265 corresponding to the states in the light orange area are shown in Fig. 5.10 when the current time step is 221, 224, and 227.

and are constant.

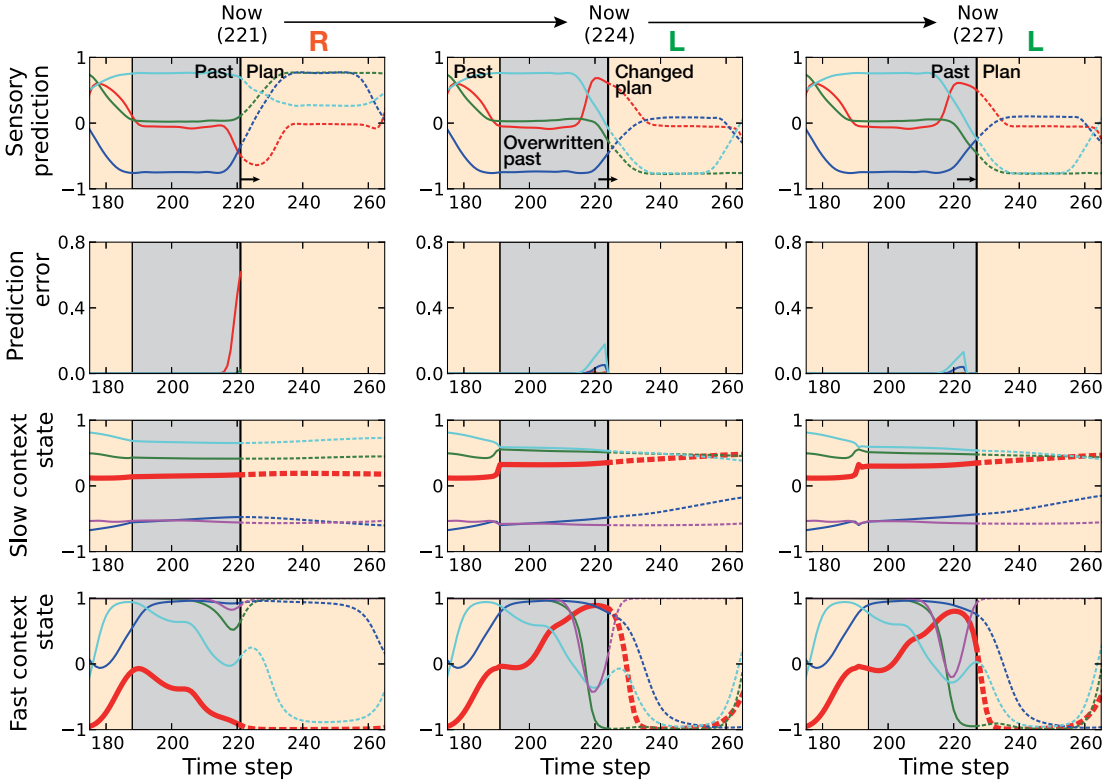


Figure 5.10 Regression dynamics. Extracted states for time steps 175 to 265 corresponding to the states in the light orange area in Fig. 5.9 (here variance predictions are not shown). The current time steps labeled "Now" in the left hand, center, right hand panels are 221 (pre-modification phase), 224 (modification phase), and 227 (post-modification phase), respectively. Time windows are indicated by gray areas in each panel. The states, shown as solid lines, to the left of the window head correspond to past states at the current time; the states, shown as dashed lines, to the right of the window head correspond to future states at the current time. These past and future states can be overwritten and changed by the ERS at each time step. Note that the states shown in Fig. 5.9 correspond to the states at the current time step shown in this figure, which are the actual states before they are overwritten by the regression.

From Fig. 5.10, we can see that the past states were overwritten in the modification phase (center panels). In the pre-modification phase (left panels), the neural dynamics of the FC state and the predicted visuo-proprioceptive states corresponded to the action primitive labeled as "R". At this time, the prediction error of the visual input representing the horizontal position of the object (red line) has increased, meaning that there was a discrepancy between the prediction and actual sensory (visual) feedback. In the previous subsection, we observed that the prediction error at each branching point cannot be suppressed (see the

wide IS distribution condition in Fig. 5.6) by only the received sensory inputs. On the other hand in the modification phase shown in Fig. 5.10, the large prediction error was suppressed by the bottom-up recognition using ERS that revises the internal neural dynamics by back-propagating the precision-weighted prediction error to the higher-level network. During this process, the internal states of the SC units at the window tail were slightly modulated, and the modulation affected the lower-level dynamics of the FC units. By comparing the pre-modification and modification phases, we can confirm that the activity of the FC units was distinctly different. By revising the past SC and FC dynamics, the previously generated prediction (past) states were overwritten, and the future plan changed from moving the hand to the right to moving it to the left. At this time, the past states outside of the window were not affected by the recognition dynamics as mentioned before. In the absence of the prediction error, the modulation of the internal neural dynamics and overwriting of past states were not performed, and forward dynamics were smoothly generated, as seen in the post-modification phase (right panels).

5.3.5 Effect of ERS on Reaction Times

We now consider the effect of error regression on reaction times. Mean reaction times over the 10 sample networks trained with the wide IS distribution condition were computed, each of which generated 32 sequences including all combinations of the five transitions of the two action primitives with and without the error regression mechanism. For computing the reaction times, the initial state optimized for the "LLRRL" sequence was adopted regardless of the other-robot's action sequences in the same manner as for the non-corresponding IS case shown in Fig. 5.7. As shown in Fig. 5.11, when the self-robot relied on only received sensory inputs (no error regression), the mean time for reaction to unanticipated external situations was 36.64 time steps (gray bar). By introducing the additional bottom-up recognition mechanism using ERS, the mean reaction time was reduced to just 8.22 time steps (red bar) with $SD = 2.75$. There is a significant difference between these mean reaction times ($t(18) = 8.54$, $p < 0.001$). This change shows that the internal contextual dynamics can be revised by means of interactions between the top-down intentional prediction and the bottom-up recognition of the actual behavior when deterministic predictive dynamics is used internally.

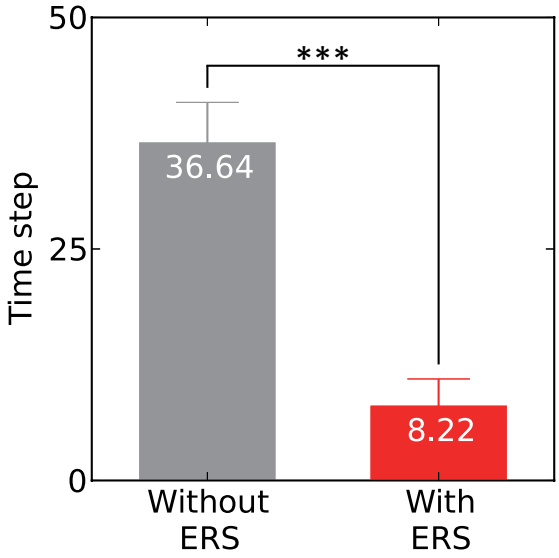


Figure 5.11 Reaction times of the self-robot during action generation with and without error regression scheme (ERS). Times are given for the results of the simulated self-robot's action generation using the network trained with the wide IS distribution condition. An arbitrarily selected initial state (values optimized for the "LLRRL" sequence) was given to the network regardless of the other-robot's action sequences. The bars and numbers in the graph correspond to mean reaction times over 10 trained sample networks, each of which generated 32 sequences including five transitions of the two action primitives. Error bars indicate the standard deviation. Stars indicate a significant difference (***: $p < 0.001$). The gray bar, for the case without ERS, corresponds to the rightmost bar in Fig. 5.7.

The mean reaction time in the wide IS distribution condition with ERS is significantly shorter ($t(18) = 5.57$, $p < 0.001$) than that in the narrow IS distribution condition (17.80 time steps shown in Fig. 5.7). This difference can be attributed to the time steps required for the mechanism of each flexible behavior generation. In the narrow IS distribution condition, the flexible primitive selection for adapting to the other-robot's behavior is based on the received sensory (especially visual) inputs that gradually change the internal neural dynamics with the longer period than the other. In contrast, in the wide IS distribution condition with ERS, the flexible selection is based on the forcible modification of the top-down prior which rapidly changes the internal neural dynamics in a discontinuous manner (see Fig. 5.9) with the shorter period than the other. It is noted furthermore that this modification force becomes much larger in the case of the wide IS distribution because the error divided by the smaller variance estimated is used for the modification by means of BPTT algorithm.

5.4 Discussion and Conclusions

This chapter hypothesized that different types of behavior generation for realizing flexible behavior, namely, reactive and proactive behavior generation, could be produced by a single neural mechanism depending on the learning condition. To test the hypothesis, robotics learning experiments were conducted. In the experiments, one robot, called the self-robot, equipped with the S-MTRNN was required to interact cooperatively with another robot, called the other-robot. In the experiments, the other-robot generated action sequences which were observed by the self-robot as probabilistic transitions of action primitives. The self-robot acquired an internal or generative model that was able to generate predictions of visuo-proprioceptive states as well as their uncertainty levels (in terms of variances or inverse precisions) through its own actions and perceptual experiences. The experimental results demonstrated that reactive behavior with a probabilistic prediction mechanism with high event-level uncertainty was developed when the initial precision characteristics of the higher-level network were not allowed in the learning process. In contrast, proactive behavior with a deterministic prediction mechanism with low event-level uncertainty was developed when the initial precision was allowed. Furthermore, the results also demonstrated that each behavior generation scheme required different adaptation mechanisms, namely, simple sensory reflex and error regression, for revising the internal neural dynamics when the

self-robot encountered unanticipated actions of the other-robot. In the following, we discuss the difference between the probabilistic model and the deterministic one by focusing on their learning capabilities and on their contributions to the development of different behavior generation schemes.

5.4.1 Treating Event-Level Uncertainty in Probabilistic or Deterministic Manner

It was demonstrated that the difference in the distribution of initial states of the higher-level network affected the learning of visuo-proprioceptive sequences observed as probabilistic transitions. When the S-MTRNN was trained with the narrow IS distribution condition, various combinatorial sequences were produced by stochastic dynamics with closed-loop generation in which self-generated noise with the estimated variances at each branching point (event-level uncertainty) determined the next primitive, as in Markov chains [98] (see Fig. 5.5A). On the other hand, when the network was trained with the wide IS distribution condition, all the learned sequences could be reproduced exactly by the top-down deterministic dynamics without any event-level uncertainty determined by the optimized initial states of the higher-level network [44, 74, 99] (see Fig. 5.5B).

The distinct models developed from the proposed S-MTRNN can be mechanized by means of a learning scheme using the maximization of a model likelihood in which sensory prediction error is divided by the predicted variance or weighted by the predicted precision. In a previous study using the conventional MTRNN model (without a variance prediction mechanism), Nishimoto et al. [99] demonstrated that the developmental process of the functional hierarchy emerged from the multiple timescale dynamics of the network and the learning scheme of prediction error minimization. During the process, a set of reusable primitives was first acquired in the lower-level network with fast dynamics. Then, sequential combinations of the primitives were acquired in the higher-level network with slow dynamics using initial precision characteristics to minimize prediction errors. By utilizing these mechanisms, Namikawa et al. [44] demonstrated that nondeterministic or probabilistic transition sequences can be embedded in deterministic chaotic dynamics, which were self-organized in the higher-level network, as pseudo-stochastic sequences. The proposed S-MTRNN (which includes a variance prediction mechanism), however, has another pathway to represent probabilistic event sequences. In short, the S-MTRNN can represent the proba-

bilistic characteristics of event transitions by means of estimating the variance of the noise externally added to the output units. This means that if the network regards observed sequences as probabilistic transitions of primitives, the network ceases to minimize prediction error at a certain level and instead optimizes the variance. This has been identified by Friston [28, 66] as the mechanism of attention that controls the acquisition of prediction error or shapes precision-weighted prediction error. However, the uniqueness of the current study is that, if the network regards the same sequences as deterministic sequential combinations of the primitives, the network tries to minimize both the prediction error and variance by attributing the potential unpredictability to deterministic chaos generated by sensitivity to initial conditions. The importance is that in the former case a probabilistic model with high event-level uncertainty is developed and in the latter case a deterministic model with low event-level uncertainty is developed, even though the same network and the same training sequences were employed.

5.4.2 Reactive Behavior versus Proactive Behavior

In an actual action generation test, we confirmed that the difference in the modeling of observed perceptual events was essential for the development of behavior generation schemes. When the S-MTRNN trained with the narrow IS distribution condition (probabilistic model) was employed, the self-robot generated reactive behavior. On the other hand, when the S-MTRNN trained with the wide IS distribution condition (deterministic model) was employed, the robot generated proactive behavior. These results originating from the different parameter settings are in general agreement with simulations of active inference [10]. Both in the proposed approach and in active inference, because action generation can be understood as fulfilling predictions (prior expectations) about proprioceptive states, the type of generative model induced by the parameter setting works as an essential factor for the development of behavior generation schemes.

During the generation of the reactive behavior, the network predicted a large variance (or uncertainty) and generated a small prediction error at each branching point (see Fig. 5.6A). These results indicate that the network predicted a neutral sensory state at branching points. Therefore, there is no difference between reaction times of the self-robot for the corresponding and non-corresponding IS cases (see the narrow IS distribution condition in Fig. 5.7). It is believed that the reactive behavior resulted from the allowance of uncertainty for sensory pre-

dictions and the development of sensitive sensory structures at each branching point.

In contrast, during the generation of the proactive behavior, the network tried to generate exact sensory predictions with almost zero variance (or uncertainty). In this case, when the intention of the self-robot corresponded to that of the other-robot, cooperative interactions could be achieved smoothly without any-time delay (see the corresponding IS case in the wide IS distribution condition in Fig. 5.7). However, when a discrepancy between their intentions occurred, inflexible behavior by the self-robot was observed (see Fig. 5.6B) and long reaction times were required (see the non-corresponding IS case in the wide IS distribution condition in Fig. 5.7). In the previous subsection, the importance of the multiple timescale dynamics for developing a functional hierarchy was discussed. From the viewpoint of learning, the slowly changing higher-level dynamics are essential for reproducing combinatorial sequences in a deterministic manner. While at the same time, the slow dynamics that are not affected by sensory inputs directly have a negative effect on generating flexible reactive behavior. As an alternative mechanism to the sensory reflex, a novel ERS, which can be considered as an extension of the predictive coding schemes used in RNNPB [8, 59] and active inference [10,11,29], was applied for revising the slow dynamics in a forcible manner by means of propagating precision-weighted prediction errors from the lower-level to the higher-level network. It should be noted that the aforementioned attention mechanism for controlling the acquisition of prediction error is utilized in this error regression process.

In an action generation test with ERS, it was confirmed that the self-robot controlled by the S-MTRNN trained with the wide IS distribution was able to generate cooperative behavior flexible when the robot encountered unanticipated actions by the other-robot. This adaptation was achieved because the ERS slightly modulated the neural activity of the higher-level network in order to maximize the model likelihood or to minimize precision-weighted prediction error. In Figs. 5.9 and 5.11, we can see the effect of the ERS on the revision of internal neural activity and on the reaction time of the self-robot, respectively. These results indicate the importance of the interaction between the top-down process for anticipating future states and the bottom-up process for recognizing perceptual reality during proactive behavior generation [8, 100]. This interactive process with ERS corresponds to the error monitoring process that might be mediated by the parietal cortex [101].

Different types of computational models have been employed to implement the robot control architectures for reactive behavior [47, 102] and proactive behavior [8, 44]. In contrast, in this study it has been demonstrated that these different behavioral schemes can be produced by a single neural mechanism. Because the learned action sequences were simple and each behavioral scheme was separately developed depending on the learning condition in this study, our next step is to consider these aspects as detailed in the next subsection.

5.4.3 Limitation and Future Work

Several issues remain to be examined in future studies. In this chapter, we have focused on the learning of visuo-proprioceptive sequences observed as probabilistic transitions, under the two distinct learning conditions of the narrow and wide IS distributions. In a set of visuo-proprioceptive sequences used for the training, all the primitives were clean and concatenated with equal probability. In other words, we have not considered a situation where each primitive includes fluctuations as demonstrated in the preceding chapter or a situation where a certain statistical bias (e.g. more movements to the left than those to the right) is given for a set of sequences [44]. Through the experiments, the robustness of the proposed model to the uncertain situational changes (probabilistic transitions of perceptual events) has been demonstrated by means of reactive behavior in the narrow IS distribution condition and proactive behavior with ERS in the wide IS distribution condition.

We should also consider the aforementioned situations for further evaluation of the proposed schemes in future study. In the context of the IS distribution, a distribution with an intermediate variance between the two conditions or a mixture distribution has not been employed. These considerations might be important to discuss the adaptive modulation between reactive behavior and proactive behavior, each of which was separately developed depending on the two distinct learning conditions in this study. Future work should therefore include follow-up work designed to evaluate these cases.

In the present study for simplicity, only the self-robot was required to change its actions and the other-robot's behavior was automatically controlled to generate fixed action sequences that were not affected by the behavior of the self-robot. That is, the interaction was unidirectional not bidirectional. However, bidirectionality in social interactions is essential for understanding the mechanism of

turn-taking behavior [103] and the acquisition of "nested" internal models in which an internal model includes itself [104, 105]. Therefore, in future studies, cases of mutual interactions by implementing the proposed model on two interacting robots will be examined. This will allow us to investigate autonomous mechanisms for the formation and manipulation of communicative interactions between cognitive agents.

Chapter 6

Conclusions

6.1 Overall Summary of the Current Study

This thesis proposed a novel computational framework for RNNs that enables intelligent robots to achieve adaptive and flexible behavior which can be preformed in real environment. The proposed framework can learn to predict both the mean and the variance of the next state of target data with fluctuations, where the variance corresponds to the uncertainty of target variables and the reciprocal of the variance is called precision. The novel RNN-based models including S-CTRNN and S-MTRNN based on the predictive learning with uncertainty estimation were evaluated through a series of numerical experiments and applied in robot learning problems.

In the series of numerical experiments, the learning, reproduction, and recognition capabilities of the proposed S-CTRNN were evaluated. The experimental results demonstrated that the S-CTRNN can learn and reproduce various types of fluctuating temporal sequences by extracting their latent stochastic structures including multiple time-invariant uncertainty, time-varying uncertainty, and state-dependent uncertainty. The results also showed that the S-CTRNN can recognize multiple fluctuating temporal sequences by inferring initial internal states of the context units which can reproduce the given sequences. These capabilities were realized by estimating uncertainty by means of the variance prediction mechanism. In both the learning and recognition processes, the predicted variance contributed to the autonomous scaling of the prediction error whose magnitude depends on the level of uncertainty included in fluctuating target sequences. The predicted variance also contributed to the reproduction process in which Gaussian noise with the predicted variance was added during the closed-loop generation.

In the first robot experiment, the S-CTRNN was applied into a framework for integrative learning of visuo-proprioceptive states. The experiment focused on the development of adaptive behavior by extracting stochastic structures latent in fluctuating behavioral (or visuo-proprioceptive) sequences with trajectory-level uncertainty. The results of the robot experiment on learning reaching movement via the kinesthetic teaching demonstrated that the S-CTRNN can reproduce stochastic structures latent in tutored fluctuating behavior sequences. Such structures can be produced not only in learned but also in unlearned situations by utilizing the generalization abilities of the trained network. In addition to the generalization abilities, adaptation to sudden changes in the target object position was also realized. These abilities were enabled by the integration of visual and proprioceptive states by means of predictive learning with uncertainty estimation.

In the second robot experiment, it was hypothesized that the underlying mechanisms for generating reactive behavior and proactive behavior could be accounted for by a single model by changing its learning conditions. For the purpose of validating this hypothesis, experiments involving a robot learning to develop cooperative interactions with another robot were conducted by utilizing S-MTRNN. This model is characterized by its capability to learn to predict complex perceptual sequences by estimating event-level uncertainty by means of self-organizing temporal hierarchy. The experimental results showed that the network dynamics were developed to generate reactive behavior with probabilistic estimation with high event-level uncertainty of subsequent behavior when the initial sensitivity characteristics in the intention state were not utilized in the learning process. In contrast, proactive behavior with deterministic prediction with low event-level uncertainty of subsequent behavior was developed when the initial sensitivity was utilized. It was concluded that the former case resulted in the development of a probabilistic structure, whereas the latter case in a deterministic one. Follow-up experiments on cooperative behavior generation examined how internal neural activity in the context units can be flexibly re-situated to the behavioral context after perceiving an unexpected behavior produced by the other in these two cases. In the network developed with a probabilistic structure, the behavioral context was re-situated by adaptation of the internal neural dynamics by means of simple sensory reflex. On the other hand, in the network developed with a deterministic structure, the behavioral context was necessary to be re-situated by means of the error regression of the internal neural activity.

6.2 Future Work

6.2.1 Action Generation for Changing the World

The studies described in this thesis did not consider the possibility that robots can change the world by acting on it. It has been described how the proposed S-CTRNN and S-MTRNN can learn to generate and recognize visuo-proprioceptive sequences under the principle of model likelihood maximization which is formally equivalent to the principle of free energy minimization [28, 60] (because the free energy is an upper bound on the negative logarithm of model likelihood). In particular, during action generation, the likelihood can be maximized by changing both the prediction state and the sensory signals that shape prediction error. In Chapter 5, for example, because the object was held by the other-robot, the self-robot was unable to change its visual input in response to its predictions. However, by changing the experimental setup, active sensory sampling can be conducted by the self-robot. For example, when the robot encounters an unanticipated situation, it can change the prediction to fit the received sensory signals and change the sensory signals to fit the prediction for minimizing prediction error. As noted in [10, 11, 28, 29], action is the only way to change sensory signals for error minimization, and thus active sampling should be considered.

6.2.2 Online Learning

The proposed S-CTRNN and S-MTRNN was trained in an offline manner using the gradient ascent method with BPTT. One might assume that the usage of the offline learning method limits the utility of the proposed scheme for more practical applications. We consider that the offline learning corresponds to the consolidation learning [106, 107] that enables cognitive agents to consolidate perceptual experiences into a long-term memory. In addition to the scheme of the offline learning and online adaptive behavior generation demonstrated in this thesis, the aspect of one-shot learning or online learning should also be considered.

6.2.3 Scalability of the Proposed Framework

Section 3.2 (Chapter 3) demonstrated that the S-CTRNN can learn to memorize 12 fluctuating Lissajous curves. Additional follow-up experiments confirmed that 144 patterns with different noise variance can also be learned by the S-CTRNN. Although these patterns include different level of noise variance and the learning

problem is not so easy, these patterns are just two-dimensional artificial data. Future work therefore should evaluate the scalability of the proposed framework by conducting learning experiments by using more realistic data such as a high-dimensional data set of multiple human or robot action patterns.

The results in Chapters 4 and 5 can be considered as the integration of exteroceptive (or visual) states and proprioceptive states of a humanoid robot by means of predictive learning with uncertainty estimation. From the point of view of the integrative learning, the scalability of the proposed framework can also be evaluated through the integrative learning of neural and behavioral data, and that of driving data, each of which contributes to the development of brain-machine interaface and autonomous car technology, respectively.

6.2.4 Beyond Uncertainty in Observable States

The proposed computational framework considers uncertainty only in observable states or target states. Natural extension of the framework is to introduce uncertainty into both the observable and hidden or context states. This enables RNNs to represent greater diversity of uncertainty. There are some possible methods. One method is to simply assume probability distribution such as Gaussian distribution for both target states and context states. However, this method may be difficult to be trained because basically RNN-based models require a large number of context units and achieving proper distribution for each unit using the gradient method is not realistic. Another method is to concatenate the S-CTRNNs (with PB) and developmentally train them from the lower to the higher levels. In this method, one level higher S-CTRNNPB is trained to predict the mean and variance states of the PB dynamics of the one level lower S-CTRNNPB. The lowest S-CTRNNPB is trained to predict the mean and variance states of fluctuating target states. Because each network only needs to estimate the mean and variance states of the dynamics of small number of PB units in the lower level, this method may be more possible approach than the former.

References

[1] Hiroyasu Iwata and Shigeki Sugano. Design of human symbiotic robot TWENDY-ONE. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pages 580–586, Kobe, may 2009. IEEE.

[2] Jonathan Bohren, Radu Bogdan Rusu, E. Gil Jones, Eitan Marder-Eppstein, Caroline Pantofaru, Melonee Wise, Lorenz Mosenlechner, Wim Meeussen, and Stefan Holzer. Towards autonomous robotic butlers: Lessons learned with the PR2. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, pages 5568–5575, Shanghai, 2011.

[3] Kunimatsu Hashimoto, Fuminori Saito, Takashi Yamamoto, and Koichi Ikeda. A field study of the human support robot in the home environment. In *Proceedings of the 2013 IEEE Workshop on Advanced Robotics and its Social Impacts*, pages 143–150, Tokyo, 2013.

[4] Lana B. Karasik, Catherine S. Tamis-Lemonda, and Karen E. Adolph. Transition from crawling to walking and infants' actions with objects and people. *Child Development*, 82(4):1199–1209, 2011.

[5] A. Meltzoff and M. Moore. Imitation of facial and manual gestures by human neonates. *Science*, 198(4312):74–78, oct 1977.

[6] M Tomasello and M J Farrar. Joint attention and early language. *Child development*, 57(1):1454–1463, 1986.

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

[8] Jun Tani. Learning to generate articulated behavior through the bottom-up and top-down interaction processes. *Neural Networks*, 16(1):11–23, 2003.

[9] Masato Ito and Jun Tani. On-line imitative interaction with a humanoid robot using a dynamic neural network model of a mirror system. *Adaptive Behavior*, 12(2):93–115, 2004.

[10] Karl J Friston, Jean Daunizeau, James Kilner, and Stefan J Kiebel. Action and behavior: A free-energy formulation. *Biological Cybernetics*, 102(3):227–260, 2010.

[11] Karl Friston, Jérémie Mattout, and James Kilner. Action understanding and active inference. *Biological Cybernetics*, 104(1-2):137–160, 2011.

[12] Yukie Nagai and Minoru Asada. Predictive Learning of Sensorimotor Information as a Key for Cognitive Development. In *Proceedings of the IROS 2015 Workshop on Sensorimotor Contingencies for Robotics*, 2015.

[13] P Dayan, G E Hinton, R M Neal, and R S Zemel. The Helmholtz machine. *Neural Computation*, 7(5):889–904, 1995.

[14] Daniel M Wolpert, Zoubin Ghahramani, and Michael I Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

[15] Daniel M Wolpert, R Chris Miall, and Mitsuo Kawato. Internal models in the cerebellum. *Trends in Cognitive Sciences*, 2(9):338–347, 1998.

[16] M Kawato. Internal models for motor control and trajectory planning. *Current opinion in neurobiology*, 9(6):718–27, dec 1999.

[17] John Mccarthy and Patrick J Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, pages 1–51, 1969.

[18] Patrick J Hayes. The Frame Problem and Related Problems in Artificial Intelligence. Technical report, University of Edinburgh., 1969.

[19] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42:335–346, 1990.

[20] Rodney A Brooks. Elephants Don't Play Chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[21] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, 1986.

[22] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[23] Rolf Pfeifer and Christian Scheier. *Understanding Intelligence*. The MIT Press, 1999.

[24] Minoru Asada, Karl F Macdorman, Hiroshi Ishiguro, and Yasuo Kuniyoshi. Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous Systems*, 37:185–193, 2001.

[25] Max Lungarella, Giorgio Metta, Rolf Pfeifer, and Giulio Sandini. Developmental robotics : A survey. *Connection Science*, 15(4):151–190, 2003.

[26] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida. Cognitive Developmental Robotics: A Survey. *IEEE Transactions on Autonomous Mental Development*, 1(1):12–34, may 2009.

[27] Jun Tani. Self-organization and compositionality in cognitive brains: A neurorobotics study. *Proceedings of the IEEE*, 102(4):586–605, apr 2014.

[28] Karl Friston. The free-energy principle: A rough guide to the brain? *Trends in Cognitive Sciences*, 13(7):293–301, 2009.

[29] Karl J Friston, Jean Daunizeau, and Stefan J Kiebel. Reinforcement learning or active inference? *PloS One*, 4(7):e6421, 2009.

[30] 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, apr 2007.

[31] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. Robot Programming by Demonstration. In *Springer Handbook of Robotics*, pages 1371–1394. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[32] Michael A Arbib, Peter Erdi, and Alice Szentagothai. *Neural Organization: Structure, Function, and Dynamics*. The MIT Press, Cambridge, MA, 1997.

[33] Michael A Arbib. Schema theory. In Michael A Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, volume 2, pages 830–834. The MIT Press, Cambridge, MA, 2002.

[34] Y. Kuniyoshi and A. Nagakubo. Humanoid as a research vehicle into flexible complex interaction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot and Systems*, volume 2, Grenoble, 1997.

[35] Tomoyuki Yamamoto and Yasuo Kuniyoshi. Stability and controllability in a rising motion: a global dynamics approach. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2467–2472, Lausanne, 2002.

[36] Yasuo Kuniyoshi, Yoshiyuki Ohmura, Koji Terada, Akihiko Nagakubo, Shin'ichiro Eitoku, and Tomoyuki Yamamoto. Embodied basis of invariant features in execution and perception of whole-body dynamic actions—knacks and focuses of Roll-and-Rise motion. *Robotics and Autonomous Systems*, 48(4):189–201, 2004.

[37] Nathaniel D Daw, Yael Niv, and Peter Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8(12):1704–1711, 2005.

[38] Mehdi Khamassi and Mark D Humphries. Integrating cortico-limbic-basal ganglia architectures for learning model-based and model-free navigation strategies. *Frontiers in Behavioral Neuroscience*, 6(79):1–19, 2012.

[39] Erwan Renaudo, Benoît Girard, Raja Chatila, and Mehdi Khamassi. Design of a control architecture for habit learning in robots. In Nathan F. Leopra, Anna Mura, Holger G Krapp, Paul F M J Verschure, and Tony J. Prescott, editors, *Biomimetic and Biohybrid Systems*, pages 249–260. Springer International Publishing AG, Cham (ZG), Switzerland, 2014.

[40] Marcel Brass and Patrick Haggard. The what, when, whether model of intentional action. *The Neuroscientist*, 14(4):319–325, 2008.

[41] Brian J White, Dirk Kerzel, and Karl R Gegenfurtner. Visually guided movements to color targets. *Experimental Brain Research*, 175(1):110–126, 2006.

[42] Andrew E Welchman, James Stanley, Malte R Schomers, R Chris Miall, and Heinrich H Bülthoff. The quick and the dead: When reaction beats intention. *Proceedings of the Royal Society B: Biological Sciences*, 277(1688):1667–1674, 2010.

[43] John R. Searle. *Intentionality: An Essay in the Philosophy of Mind*. Cambridge University Press, New York, NY, 1983.

[44] Jun Namikawa, Ryunosuke Nishimoto, and Jun Tani. A neurodynamic account of spontaneous behaviour. *PLoS Computational Biology*, 7(10):e1002221, 2011.

[45] Jun Tani and Naohiro Fukumura. Learning Goal-Directed Sensory-Based Navigation of a Mobile Robot. *Neural Networks*, 7(3):553–563, 1994.

[46] Randall D Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4), dec 1995.

[47] Rolf Pfeifer and Christian Scheier. Sensory—motor coordination: The metaphor and beyond. *Robotics and Autonomous Systems*, 20(2-4):157–178, 1997.

[48] Jun Tani. Model-based learning for mobile robot navigation from the dynamical systems perspective. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 26(3):421–436, jan 1996.

[49] Michael I. Jordan and David E. Rumelhart. Forward Models: Supervised Learning with a Distal Teacher. *Cognitive Science*, 16(3):307–354, jul 1992.

[50] Norikazu Sugimoto, Jun Morimoto, Sang-Ho Hyon, and Mitsuo Kawato. The eMOSAIC model for humanoid robot control. *Neural Networks*, 29-30:8–19, 2012.

[51] Fady Alnajjar, Yuichi Yamashita, and Jun Tani. The hierarchical and functional connectivity of higher-order cognitive mechanisms: neurorobotic

model to investigate the stability and flexibility of working memory. *Frontiers in neurorobotics*, 7(February):2, jan 2013.

[52] Karl Friston. Hierarchical models in the brain. *PLoS Computational Biology*, 4(11):e1000211, 2008.

[53] R P Rao and D H Ballard. Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, 1999.

[54] Andy Clark. Whatever next? Predictive brains, situated agents, and the future of cognitive science. *The Behavioral and Brain Sciences*, 36(3):181–204, 2013.

[55] Michael I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, pages 531–546, Amherst, MA, aug 1986.

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

[57] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[58] Jordan B. Pollack. The induction of dynamical recognizers. *Machine Learning*, 7(2-3):227–252, 1991.

[59] 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(3):323–337, 2006.

[60] Karl Friston. The free-energy principle: A unified brain theory? *Nature reviews. Neuroscience*, 11(2):127–38, 2010.

[61] Jakob Hohwy. *The Predicrive Mind*. Oxford University Press, Oxford, 2013.

[62] Jun Tani and M Ito. Self-organization of behavioral primitives as multiple attractor dynamics: A robot experiment. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 33(4):481–488, jul 2003.

[63] David C Knill and Alexandre Pouget. The Bayesian brain: The role of uncertainty in neural coding and computation. *Trends in Neurosciences*, 27(12):712–719, 2004.

[64] Karl Friston. A theory of cortical responses. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 360(1456):815–836, 2005.

[65] Karl Friston, James Kilner, and Lee Harrison. A free energy principle for the brain. *Journal of Physiology*, 100(1-3):70–87, 2006.

[66] Harriet Feldman and Karl J Friston. Attention, uncertainty, and free-energy. *Frontiers in Human Neuroscience*, 4(215):1–23, 2010.

[67] Jakob Hohwy. Attention and conscious perception in the hypothesis testing brain. *Frontiers in Psychology*, 3(96):1–14, 2012.

[68] Hanneke E M den Ouden, Peter Kok, and Floris P de Lange. How prediction errors shape perception, attention, and motivation. *Frontiers in Psychology*, 3(548):1–12, 2012.

[69] Andy Clark. *Surfing Uncertainty: Prediction, Action, and the Embodied Mind*. Oxford University Press, 2015.

[70] Jun Tani and Naohiro Fukumura. Embedding a grammatical description in deterministic chaos: An experiment in recurrent neural learning. *Biological Cybernetics*, 72(4):365–370, 1995.

[71] Jun Namikawa and Jun Tani. Learning to imitate stochastic time series in a compositional way by chaos. *Neural Networks*, 23(5):625–638, 2010.

[72] J M Fuster. The prefrontal cortex–an update: Time is of the essence. *Neuron*, 30(2):319–333, 2001.

[73] Matthew M Botvinick. Hierarchical models of behavior and prefrontal function. *Trends in Cognitive Sciences*, 12(5):201–208, 2008.

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

[75] Aude Billard, Yann Epars, Sylvain Calinon, Stefan Schaal, and Gordon Cheng. Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, 47(2-3):69–77, jun 2004.

[76] S Calinon, F Guenter, and A Billard. Goal-Directed Imitation in a Humanoid Robot. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, number April, pages 299–304. IEEE, 2005.

[77] Aude G. Billard, Sylvain Calinon, and Florent Guenter. Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems*, 54(5):370–384, may 2006.

[78] Sylvain Calinon and Aude Billard. Learning of Gestures by Imitation in a Humanoid Robot. In Kerstin Dautenhahn and Chrystopher Nehaniv, editors, *Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions*, pages 153–177. Cambridge Univ. Press, Cambridge, 2007.

[79] S Mohammad Khansari-Zadeh and Aude Billard. Imitation learning of globally stable non-linear point-to-point robot motions using nonlinear programming. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, number 2, pages 2676–2683. IEEE, oct 2010.

[80] S Mohammad Khansari-Zadeh and Aude Billard. Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[81] S.M. Khansari-Zadeh, Klas Kronander, and Aude Billard. Learning to Play Minigolf: A Dynamical System-Based Approach. *Advanced Robotics*, 26(17):1967–1993, dec 2012.

[82] Sylvain Calinon, Zhibin Li, Tohid Alizadeh, Nikos G Tsagarakis, and Darwin G Caldwell. Statistical dynamical systems for skills acquisition in humanoids. In *Proceedings of the 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 323–329. IEEE, nov 2012.

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

[84] Ryu Nishimoto and Jun Tani. Learning to generate combinatorial action sequences utilizing the initial sensitivity of deterministic dynamical systems. *Neural Networks*, 17(7):925–933, 2004.

[85] R. Nishimoto, J. Namikawa, and J. Tani. Learning multiple goal-directed actions through self-organization of a dynamic neural network model: A humanoid robot experiment. *Adaptive Behavior*, 16(2-3):166–181, 2008.

[86] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Stephen I Ryu, and Krishna V Shenoy. Cortical preparatory activity: Representation of movement or first cog in a dynamical machine? *Neuron*, 68(3):387–400, 2010.

[87] Kenji Doya and Shuji Yoshizawa. Adaptive neural oscillator using continuous-time back-propagation learning. *Neural Networks*, 2(5):375–385, jan 1989.

[88] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801–806, jan 1993.

[89] Jun Namikawa and Jun Tani. A model for learning to segment temporal sequences, utilizing a mixture of RNN experts together with adaptive variance. *Neural networks*, 21(10):1466–1475, 2008.

[90] Stefan J. Kiebel, Jean Daunizeau, and Karl J. Friston. A hierarchy of time-scales and the brain. *PLoS Computational Biology*, 4(11):e1000209, 2008.

[91] Uri Hasson, Janice Chen, and Christopher J. Honey. Hierarchical process memory: Memory as an integral component of information processing. *Trends in Cognitive Sciences*, 19(6):304–313, 2015.

[92] Yuichi Yamashita and Jun Tani. Spontaneous prediction error generation in schizophrenia. *PloS One*, 7(5):e37843, 2012.

[93] Sylvain Calinon and Aude Billard. Statistical Learning by Imitation of Competing Constraints in Joint Space and Task Space. *Advanced Robotics*, 23(15):2059–2076, jan 2009.

[94] Manuel Muhlig, Michael Gienger, Sven Hellbach, Jochen J Steil, and Christian Goerick. Task-level imitation learning using variance-based movement optimization. In *2009 IEEE International Conference on Robotics and Automation*, volume 67, pages 1177–1184. IEEE, may 2009.

[95] Adam Coates, Pieter Abbeel, and Andrew Y. Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 144–151, New York, New York, USA, 2008. ACM Press.

[96] E Gribovskaya, S M Khansari-Zadeh, and Aude Billard. Learning Nonlinear Multivariate Dynamics of Motion in Robotic Manipulators. *International Journal of Robotics Research*, 30(1):80–117, 2011.

[97] Tom Ziemke, Dan-Anders Jirenhed, and Germund Hesslow. Internal simulation of perception: A minimal neuro-robotic model. *Neurocomputing*, 68:85–104, 2005.

[98] Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[99] Ryunosuke Nishimoto and Jun Tani. Development of hierarchical structures for actions and motor imagery: A constructivist view from synthetic neuro-robotics study. *Psychological Research*, 73(4):545–558, 2009.

[100] Jun Tani. Autonomy of self at criticality: The perspective from synthetic neuro-robotics. *Adaptive Behavior*, 17(5):421–443, 2009.

[101] Michel Desmurget, Karen T Reilly, Nathalie Richard, Alexandru Szathmari, Carmine Mottolese, and Angela Sirigu. Movement intention after parietal cortex stimulation in humans. *Science*, 324(5928):811–813, 2009.

[102] J. Leitner, M. Frank, A. Fosrter, and J. Schmidhuber. Reactive reaching and grasping on a humanoid towards closing the action-perception loop on the iCub. In *Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics*, pages 102–109, Vienna, sep 2014.

[103] Takashi Ikegami and Hiroyuki Iizuka. Turn-taking interaction as a cooperative and co-creative process. *Infant Behavior & Development*, 30(2):278–288, 2007.

[104] Makoto Taiji and Takashi Ikegami. Dynamics of internal models in game players. *Physica D: Nonlinear Phenomena*, 134(2):253–266, 1999.

[105] Wako Yoshida, Ben Seymour, Karl J Friston, and Raymond J Dolan. Neural mechanisms of belief inference during cooperative games. *The Journal of Neuroscience*, 30(32):10744–10751, 2010.

[106] J L McClelland, B L McNaughton, and R C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995.

[107] Lynn Nadel and Morris Moscovitch. Memory consolidation, retrograde amnesia and the hippocampal complex. *Current Opinion in Neurobiology*, 7(2):217–227, 1997.

[108] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Appendix A

Details of Conventional Neural Networks

This appendix provides the details of computational framework of the conventional artificial neural networks including FNNs and RNNs that are briefly reviewed in Chapter 2 (Section 2.2).

A.1 Feedforward Neural Networks (FNNs)

Figure A.1 provides a network diagram of an FNN or detailed illustration of Fig. 2.1A in Chapter 2 (Section 2.2.1). When multiple hidden layers are used to construct a network architecture in which connections from the units in a hidden layer to those in another hidden layer are considered, such a network is called *deep neural network* (DNN) whose special learning schemes are referred to as *deep learning* [108]. For simplicity, this section considers only one hidden layer case, however, computation methods introduced later can be easily extended to those for multiple layer cases.

The remaining subsections provide the computation method of forward propagation of FNNs and their gradient-based predictive learning scheme.

A.1.1 Forward Propagation

Consider an FNN consisting of an input layer with N_{I}-dimensional units, a hidden layer with N_{H}-dimensional units, and an output layer with N_{O}-dimensional units as shown in Fig. A.1. The forward dynamics of the internal states of the ith

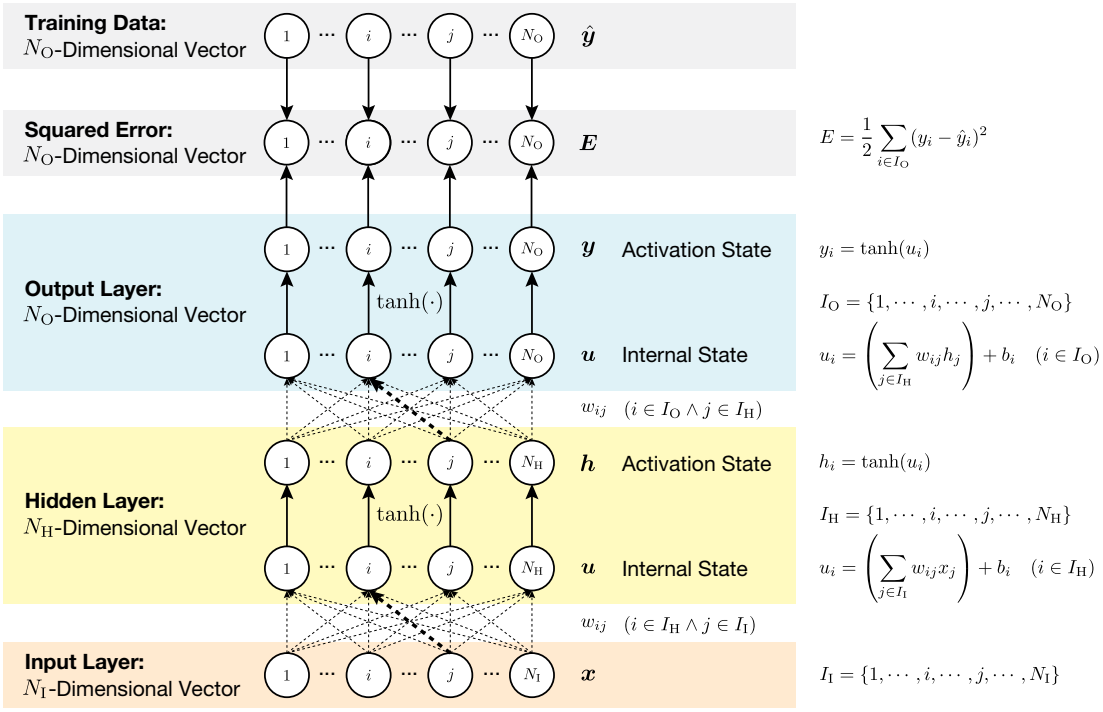


Figure A.1 Network diagram of FNN. The input layer including N_{I}-dimensional neural units (nodes in the input layer) receives external input states $\boldsymbol{x}$. These units are connected with N_{H}-dimensional neural units (nodes in the bottom of the hidden layer) and internal states $\boldsymbol{u}$ of the hidden layer are computed as an weighted sum of the current input states. Synaptic weights are represented by links (black dashed lines) between the nodes. The bold black dashed line indicates the connection from the jth to the ith unit (w_{ij}). The computed internal states are transformed to activation states $\boldsymbol{h}$ (nodes in the top of the hidden layer) by using non-linear function such as $\tanh(\cdot)$ used in the figure. In a similar way, internal states $\boldsymbol{u}$ of the output layer are computed as an weighted sum of hidden activation states and output activation states $\boldsymbol{y}$ are achieved by applying the non-linear function to the internal states. By computing the difference between output activation states $\boldsymbol{y}$ and target states $\hat{\boldsymbol{y}}$, we obtain prediction error defined by squared error $\boldsymbol{E}$ that is used for the gradient-based predictive learning with BP. Equations corresponding to each process are written in the right side of the figure.

hidden and output units (u_i) are given by

$$u_i = \begin{cases} \left(\sum_{j \in I_{\mathrm{I}}} w_{ij} x_j \right) + b_i & (i \in I_{\mathrm{H}}), \\ \left(\sum_{j \in I_{\mathrm{H}}} w_{ij} h_j \right) + b_i & (i \in I_{\mathrm{O}}), \end{cases} \tag{A.1}$$

where

$I_{\mathrm{I}}, I_{\mathrm{H}}, I_{\mathrm{O}}$: index sets for the input, hidden, and output units, respectively (e.g. $I_{\mathrm{I}} = \{1, \cdots, i, \cdots, j, \cdots, N_{\mathrm{I}}\}$),

w_{ij}: synaptic weight of the connection from the jth to the ith unit,

x_j: the jth external input state,

h_j: neural activation state of the jth hidden unit,

b_i: bias of the ith unit.

The neural activation states of hidden units h_i and output units y_i are calculated by using the corresponding activation functions $f(\cdot)$ and $g(\cdot)$ as follows:

$$h_i = f(u_i) \qquad (i \in I_{\mathrm{H}}), \tag{A.2}$$

$$y_i = g(u_i) \qquad (i \in I_{\mathrm{O}}). \tag{A.3}$$

For $f(\cdot)$ and $g(\cdot)$, the hyperbolic tangent, sigmoid, softmax, and linear functions are used depending on the problem to be solved. In what follows, as an example, the hyperbolic tangent $\tanh(\cdot)$ is used for both $f(\cdot)$ and $g(\cdot)$.

A.1.2 Predictive Learning

The predictive learning (parameter optimization) of FNNs is conducted based on the prediction error minimization using the gradient descent method. In this subsection, the definition of objective function, gradient descent method, and BP algorithm are introduced.

Objective Function

Suppose that we are given training data comprising D sets of input–target observations $\{(\boldsymbol{x}^{(1)}, \hat{\boldsymbol{y}}^{(1)}), (\boldsymbol{x}^{(2)}, \hat{\boldsymbol{y}}^{(2)}), \ \ldots \ (\boldsymbol{x}^{(d)}, \hat{\boldsymbol{y}}^{(d)}), \ \ldots \ (\boldsymbol{x}^{(D)}, \hat{\boldsymbol{y}}^{(D)})\}$, where $d = \{1, 2, \cdots, d, \cdots, D\}$ is the index of the observations or data points, $\boldsymbol{x}^{(d)}$ is

the dth N_{I}-dimensional input states, and $\hat{\boldsymbol{y}}^{(d)}$ is the dth N_{O}-dimensional target states. If the input states are sampled with a certain sampling rate or time step from the states of a dynamical system and if the target states satisfy $\hat{\boldsymbol{y}}^{(d)} = \boldsymbol{x}^{(d+1)}$, the problem can be considered as a predictive learning problem. In this situation, the objective function (prediction error defined by squared error) E is given by

$$E = \sum_{d=1}^{D} E^{(d)}, \tag{A.4}$$

$$E^{(d)} = \sum_{i \in I_{\mathrm{O}}} E_i^{(d)} \tag{A.5}$$

$$= \sum_{i \in I_O} \frac{(\hat{y}_i^{(d)} - y_i^{(d)})^2}{2}, \tag{A.6}$$

where $\hat{y}_i^{(d)}$ and $y_i^{(d)}$ are the ith dimension of the target states $\hat{\boldsymbol{y}}^{(d)}$ and that of the output states $\boldsymbol{y}^{(d)}$, respectively.

Gradient Descent Method

The network parameters, synaptic weights w_{ij} and biases b_i that are collected by $\boldsymbol{\theta}$, are optimized to minimize the prediction error (A.4) by using the gradient descent method as follows:

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \alpha \Delta \boldsymbol{\theta}_n, \tag{A.7}$$

$$\Delta \boldsymbol{\theta}_n = -\frac{\partial E(\boldsymbol{\theta}_{n-1})}{\partial \boldsymbol{\theta}} + \eta \Delta \boldsymbol{\theta}_{n-1}, \tag{A.8}$$

$$\Delta \boldsymbol{\theta}_0 = 0, \tag{A.9}$$

where n is the learning step, α is the learning rate, and η is a parameter representing the momentum term.

Back-Propagation

Gradients of the objective function with respect to parameters w_{ij} and b_i can be obtained by the BP method. In the following, each variable corresponds to the dth target states in training data is represented as $(\cdot)^{(d)}$.

Because $E^{(d)}$ depends on the w_{ij} and b_i through the internal state $u_i^{(d)}$, each

gradient is given by

$$\frac{\partial E^{(d)}}{\partial w_{ij}} = \frac{\partial E^{(d)}}{\partial u_i^{(d)}} \frac{\partial u_i^{(d)}}{\partial w_{ij}} = \begin{cases} \delta_i^{(d)} x_j^{(d)} & (i \in I_{\mathrm{H}} \wedge j \in I_{\mathrm{I}}), \\ \delta_i^{(d)} h_j^{(d)} & (i \in I_{\mathrm{O}} \wedge j \in I_{\mathrm{H}}), \end{cases} \tag{A.10}$$

$$\frac{\partial E^{(d)}}{\partial b_i} = \frac{\partial E^{(d)}}{\partial u_i^{(d)}} \frac{\partial u_i^{(d)}}{\partial b_i} = \delta_i^{(d)} \quad (i \in I_{\mathrm{O}} \cup I_{\mathrm{H}}), \tag{A.11}$$

where

$$\delta_i^{(d)} = \frac{\partial E^{(d)}}{\partial u_i^{(d)}} = \begin{cases} \displaystyle\sum_{k \in I_{\mathrm{O}}} \frac{\partial E^{(d)}}{\partial u_k^{(d)}} \frac{\partial u_k^{(d)}}{\partial u_i^{(d)}} = \sum_{k \in I_{\mathrm{O}}} \delta_k^{(d)} \frac{\partial}{\partial u_i^{(d)}} \left(\sum_{j \in I_H} w_{k,j} h_j^{(d)} \right) \\ \qquad = \left\{ 1 - (h_i^{(d)})^2 \right\} \displaystyle\sum_{k \in I_{\mathrm{O}}} w_{ki} \delta_k^{(d)} \qquad (i \in I_{\mathrm{H}}), \\ \dfrac{\partial E^{(d)}}{\partial y_i^{(d)}} \dfrac{\partial y_i^{(d)}}{\partial u_i^{(d)}} = -\left\{ 1 - (y_i^{(d)})^2 \right\} (\hat{y}_i - y_i^{(d)}) \qquad (i \in I_{\mathrm{O}}). \end{cases} \tag{A.12}$$

For example, the first equation in (A.12) can be obtained by applying the chain rule under the consideration of the influence of the internal state of the ith hidden unit ($u_i^{(d)}$: $i \in I_{\mathrm{H}}$) on the prediction error $E^{(d)}$ through the internal state of the kth output unit ($u_k^{(d)}$: $k \in I_{\mathrm{O}}$). These relationships among variables can be understood from the network diagram illustrated in Fig. A.1.

It should be noted that because the gradient of the total error E with respect to each parameter corresponds to the sum of the gradients of each error $E^{(d)}$, the learning is conducted by using the following equation.

$$\frac{\partial E}{\partial \boldsymbol{\theta}} = \sum_{d=1}^{D} \frac{\partial E^{(d)}}{\partial \boldsymbol{\theta}}. \tag{A.13}$$

A.2 Recurrent Neural Networks (RNNs)

Figure A.2 provides a network diagram of an RNN or detailed illustration of Fig. 2.1B in Chapter 2 (Section 2.2.1).

The remaining subsections provide the computation method of forward propagation of RNNs and their gradient-based predictive learning scheme derived from that for FNNs.

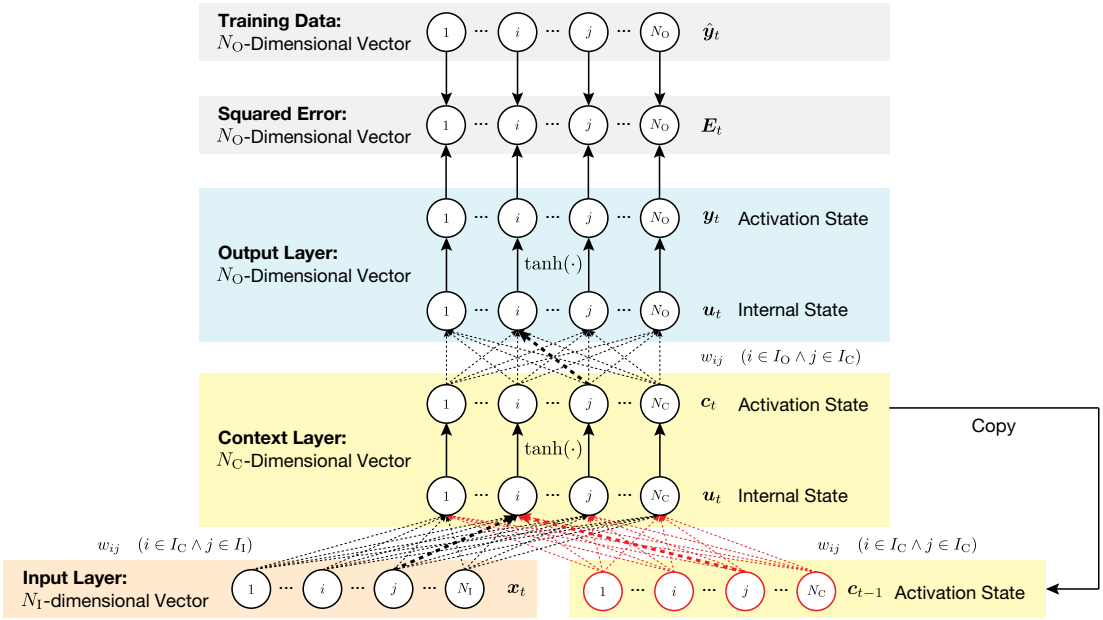


Figure A.2 Network diagram of RNN. The additional parts compared to the FNN shown in Fig. A.1 are colored with red (activation states in the context layer at the previous time step and connections from these states to the internal states in the context layer). The other parts are the same as those for FNN.

A.2.1 Forward Propagation

Consider an RNN consisting of an input layer with N_{I}-dimensional units, a context layer with N_{C}-dimensional units, and an output layer with N_{O}-dimensional units as shown in Fig. A.2. The forward dynamics of the internal states of the ith context and output unit at time step $1 \leq t$ corresponding to the sth target sequence ($u_{t,i}^{(s)}$) are given by

$$u_{t,i}^{(s)} = \begin{cases} \left(\sum_{j \in I_{\mathrm{I}}} w_{ij} x_{t,j}^{(s)}\right) + \left(\sum_{j \in I_{\mathrm{C}}} w_{ij} c_{t-1,j}^{(s)}\right) + b_i & (i \in I_{\mathrm{C}}), \\ \left(\sum_{j \in I_{\mathrm{C}}} w_{ij} c_{t,j}^{(s)}\right) + b_i & (i \in I_{\mathrm{O}}), \end{cases} \tag{A.14}$$

where

$I_{\mathrm{I}}, I_{\mathrm{C}}, I_{\mathrm{O}}$: index sets for the input, context, and output units, respectively,
w_{ij}: synaptic weight of the connection from the jth unit to the ith unit,
$x_{t,j}^{(s)}$: the jth external input state at time step t corresponding to the sth target sequence,
$c_{t,j}^{(s)}$: neural activation state of the jth context unit at time step t corresponding to the sth target sequence,
b_i: bias of the ith unit.

It should be noted that the initial states of the context units ($c_{0,i}^{(s)}$) are optimized for each sequence by which RNNs can discriminate different sequences such as multiple attractor sequences and branching sequences. This characteristic of nonlinear dynamical system is referred to as *sensitivity to initial conditions* or *initial precision characteristic*. The detail about the characteristic is explained in Chapter 2 (Section 2.2.3).

In CTRNN models [46, 87, 88], internal states of each context unit are assumed to be influenced not only by the weighted sum of current input states and previous context activation states, but also their previous internal states. This characteristic of contextual dynamics is described by the following differential equation:

$$\tau_i \dot{u}_i^{(s)} = -u_i^{(s)} + \left(\sum_{j \in I_{\mathrm{I}}} w_{ij} x_j^{(s)}\right) + \left(\sum_{j \in I_{\mathrm{C}}} w_{ij} c_j^{(s)}\right) + b_i \qquad (i \in I_{\mathrm{C}}), \tag{A.15}$$

where

τ_i: time constant of the ith context unit,
$u_i^{(s)}$: internal state of the ith unit corresponding to the sth target sequence,
$I_{\mathrm{I}}, I_{\mathrm{C}}, I_{\mathrm{O}}$: index sets for the input, context, and output units, respectively,
w_{ij}: synaptic weight of the connection from the jth unit to the ith unit,
$x_j^{(s)}$: the jth external input state corresponding to the sth targe sequence,
$c_j^{(s)}$: neural activation state of the jth context unit corresponding to the sth target sequence,
b_i: bias of the ith unit.

The above equation can be rewritten as the following numerical equation:

$$\tau_i \left(\frac{u_{t,i}^{(s)} - u_{t-1,i}^{(s)}}{\Delta t} \right) = -u_{t-1,i}^{(s)} + \left(\sum_{j \in I_{\mathrm{I}}} w_{ij} x_{t,j}^{(s)} \right) + \left(\sum_{j \in I_{\mathrm{C}}} w_{ij} c_{t-1,j}^{(s)} \right) + b_i \quad (i \in I_{\mathrm{C}}), \tag{A.16}$$

$$u_{t,i}^{(s)} = \left(1 - \frac{\Delta t}{\tau_i} \right) u_{t-1,i}^{(s)} + \frac{\Delta t}{\tau_i} \left\{ \left(\sum_{j \in I_{\mathrm{I}}} w_{ij} x_{t,j}^{(s)} \right) + \left(\sum_{j \in I_{\mathrm{C}}} w_{ij} c_{t-1,j}^{(s)} \right) + b_i \right\} \quad (i \in I_{\mathrm{C}}). \tag{A.17}$$

By considering $\tau_i / \Delta t \to \tau_i$, the internal states of the ith unit at time step $1 \leq t$ corresponding to the sth target sequence ($u_{t,i}^{(s)}$) are given by

$$u_{t,i}^{(s)} = \begin{cases} \left(1 - \dfrac{1}{\tau_i} \right) u_{t-1,i}^{(s)} + \dfrac{1}{\tau_i} \left\{ \left(\displaystyle\sum_{j \in I_{\mathrm{I}}} w_{ij} x_{t,j}^{(s)} \right) + \left(\displaystyle\sum_{j \in I_{\mathrm{C}}} w_{ij} c_{t-1,j}^{(s)} \right) + b_i \right\} & (1 \leq t \wedge i \in I_{\mathrm{C}}), \\ \left(\displaystyle\sum_{j \in I_{\mathrm{C}}} w_{ij} c_{t,j}^{(s)} \right) + b_i & (1 \leq t \wedge i \in I_{\mathrm{O}}). \end{cases} \tag{A.18}$$

By setting the time constant $\tau_i = 1$, the forward dynamics of the internal state of the context unit in (A.18) is exactly the same as those in (A.14). In other words, the RNN can be cast as the special case of CTRNNs. Therefore, in what follows, only the equations for CTRNNs are explained.

The neural activation states of context units $c_{t,i}^{(s)}$ and output units $y_{t,i}^{(s)}$ at time step t corresponding to the sth target sequence are calculated by using the activation function $\tanh(\cdot)$ as follows:

$$c_{t,i}^{(s)} = \tanh(u_{t,i}^{(s)}) \qquad (0 \leq t \wedge i \in I_{\mathrm{C}}), \tag{A.19}$$

$$y_{t,i}^{(s)} = \tanh(u_{t,i}^{(s)}) \qquad (1 \leq t \wedge i \in I_{\mathrm{O}}). \tag{A.20}$$

A network diagram of a CTRNN is illustrated in Fig. A.3.

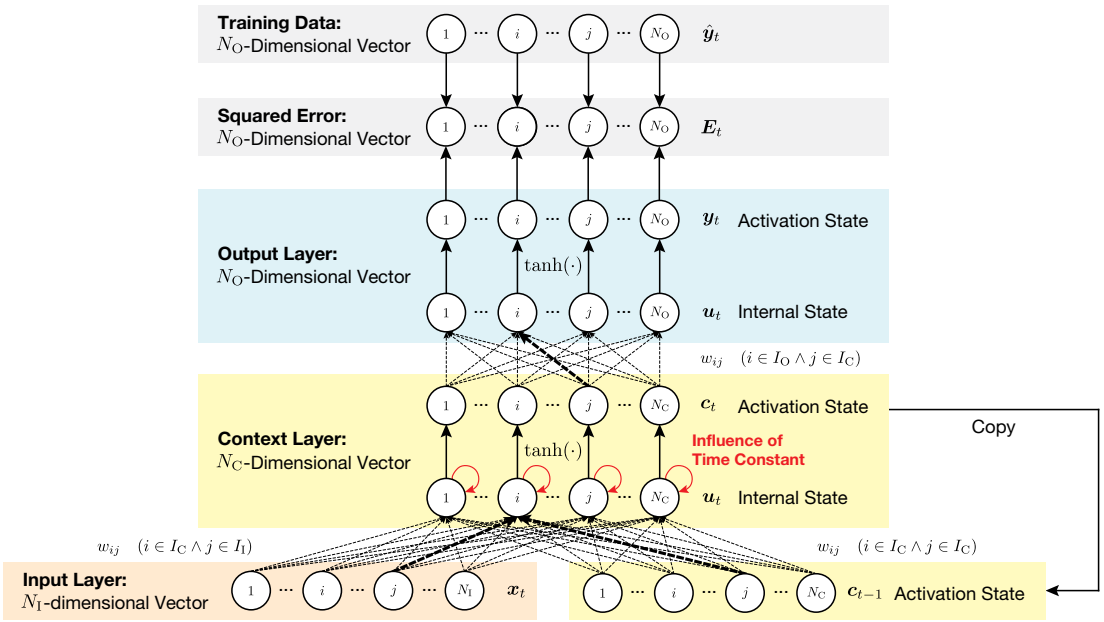


Figure A.3 Network diagram of CTRNN. The additional parts compared to the RNN shown in Fig. A.2 are colored with red (arrows representing the influence of time constants on each internal state in the context layer). The other parts are the same as those for the RNN.

A.2.2 Predictive Learning

The predictive learning of CTRNNs is conducted based on the prediction error minimization using the gradient descent method in the same manner as that of FNN. In this subsection, the definition of objective function, gradient descent method, and BPTT algorithm are introduced.

Objective Function

Suppose that we are given S target sequences, each of which consists of

$$\{\hat{\boldsymbol{y}}_t^{(s)}\}_{t=1}^{T^{(s)}} = \{\hat{\boldsymbol{y}}_1^{(s)}, \hat{\boldsymbol{y}}_2^{(s)}, \cdots, \hat{\boldsymbol{y}}_t^{(s)}, \cdots, \hat{\boldsymbol{y}}_{T^{(s)}}^{(s)}\}, \tag{A.21}$$

where s is the index of the sequence, and $T^{(s)}$ is the length of the sth sequence. Basically, the input to an CTRNN is described as $\boldsymbol{x}_t^{(s)} = \hat{\boldsymbol{y}}_{t-1}^{(s)}$. In this situation, the objective function (prediction error defined by squared error) E is given by

$$E = \sum_{s \in I_{\mathrm{S}}} \sum_{t=1}^{T^{(s)}} \sum_{i \in I_{\mathrm{O}}} \frac{(\hat{y}_{t,i}^{(s)} - y_{t,i}^{(s)})^2}{2} \tag{A.22}$$

where $I_{\mathrm{S}} = \{1, 2, \cdots, s, \cdots, S\}$ is the index set for the sequences and $\hat{y}_{t,i}^{(s)}$ is the ith dimension value of $\hat{\boldsymbol{y}}_t^{(s)}$.

Gradient Descent Method

The network parameters, synaptic weights w_{ij}, biases b_i, and initial internal states of context units for each sequence ($u_{0,i}^{(s)}$) that are collected by $\boldsymbol{\theta}$, are optimized to minimize the prediction error (A.22) by using the gradient descent method as follows:

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \alpha \Delta \boldsymbol{\theta}_n, \tag{A.23}$$

$$\Delta \boldsymbol{\theta}_n = -\frac{\partial E(\boldsymbol{\theta}_{n-1})}{\partial \boldsymbol{\theta}} + \eta \Delta \boldsymbol{\theta}_{n-1}, \tag{A.24}$$

$$\Delta \boldsymbol{\theta}_0 = 0, \tag{A.25}$$

where n is the learning step, α is the learning rate, and η is a parameter representing the momentum term.

Back-Propagation Through Time

Gradients of the objective function with respect to (time-invariant) parameters w_{ij} and b_i, and (time-varying) variables or states $u_{t,i}^{(s)}$ can be obtained by the BPTT method by considering the dependency of the error E on each parameter and state (to consider the dependency please refer to Figs. 2.1D and A.3):

$$\frac{\partial E}{\partial w_{ij}} = \begin{cases} \dfrac{1}{\tau_i} \displaystyle\sum_{s \in I_{\mathrm{S}}} \sum_{t=1}^{T^{(s)}} x_{t,j}^{(s)} \frac{\partial E}{\partial u_{t,i}^{(s)}} & (i \in I_{\mathrm{C}} \wedge j \in I_{\mathrm{I}}), \\ \dfrac{1}{\tau_i} \displaystyle\sum_{s \in I_{\mathrm{S}}} \sum_{t=1}^{T^{(s)}} c_{t-1,j}^{(s)} \frac{\partial E}{\partial u_{t,i}^{(s)}} & (i \in I_{\mathrm{C}} \wedge j \in I_{\mathrm{C}}), \\ \displaystyle\sum_{s \in I_{\mathrm{S}}} \sum_{t=1}^{T^{(s)}} c_{t,j}^{(s)} \frac{\partial E}{\partial u_{t,i}^{(s)}} & (i \in I_{\mathrm{O}} \wedge j \in I_{\mathrm{C}}), \end{cases} \tag{A.26}$$

$$\frac{\partial E}{\partial b_i} = \begin{cases} \dfrac{1}{\tau_i} \displaystyle\sum_{s \in I_{\mathrm{S}}} \sum_{t=1}^{T^{(s)}} \frac{\partial E}{\partial u_{t,i}^{(s)}} & (i \in I_{\mathrm{C}}), \\ \displaystyle\sum_{s \in I_{\mathrm{S}}} \sum_{t=1}^{T^{(s)}} \frac{\partial E}{\partial u_{t,i}^{(s)}} & (i \in I_{\mathrm{O}}), \end{cases} \tag{A.27}$$

$$\frac{\partial E}{\partial u_{t,i}^{(s)}} = \begin{cases} \left\{1 - (c_{t,i}^{(s)})^2\right\} \left(\displaystyle\sum_{k \in I_{\mathrm{C}}} \frac{w_{ki}}{\tau_k} \frac{\partial E}{\partial u_{t+1,k}^{(s)}} + \sum_{k \in I_{\mathrm{O}}} w_{ki} \frac{\partial E}{\partial u_{t,k}^{(s)}} \right) & \\ \qquad + \left(1 - \dfrac{1}{\tau_i}\right) \dfrac{\partial E}{\partial u_{t+1,i}^{(s)}} & (0 \leq t \wedge i \in I_{\mathrm{C}}), \\ -\left\{1 - (y_{t,i}^{(s)})^2\right\} (\hat{y}_{t,i}^{(s)} - y_{t,i}^{(s)}) & (1 \leq t \wedge i \in I_{\mathrm{O}}). \end{cases} \tag{A.28}$$

Appendix B

Stochastic Recurrent Neural Networks

B.1 Derivation of the BPTT

B.1.1 Supplemental Explanation for (2.20)

$$\begin{aligned}\frac{\partial \ln L_{\text{out}}}{\partial w_{ij}} &= \sum_{s\in I_{\text{S}}}\sum_{t=1}^{T^{(s)}} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}^{(s)}} \frac{\partial u_{t,i}^{(s)}}{\partial w_{ij}} \\ &= \begin{cases} \dfrac{1}{\tau_i}\displaystyle\sum_{s\in I_{\text{S}}}\sum_{t=1}^{T^{(s)}} x_{t,j}^{(s)} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}^{(s)}} & (i \in I_{\text{C}} \wedge j \in I_{\text{I}}), \\ \dfrac{1}{\tau_i}\displaystyle\sum_{s\in I_{\text{S}}}\sum_{t=1}^{T^{(s)}} c_{t-1,j}^{(s)} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}^{(s)}} & (i \in I_{\text{C}} \wedge j \in I_{\text{C}}), \\ \displaystyle\sum_{s\in I_{\text{S}}}\sum_{t=1}^{T^{(s)}} c_{t,j}^{(s)} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}^{(s)}} & (i \in I_{\text{O}} \cup I_{\text{V}} \wedge j \in I_{\text{C}}). \end{cases}\end{aligned} \tag{B.1}$$

B.1.2 Supplemental Explanation for (2.21)

$$\begin{aligned}\frac{\partial \ln L_{\text{out}}}{\partial b_i} &= \sum_{s\in I_{\text{S}}}\sum_{t=1}^{T^{(s)}} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}^{(s)}} \frac{\partial u_{t,i}^{(s)}}{\partial b_i} \\ &= \begin{cases} \dfrac{1}{\tau_i}\displaystyle\sum_{s\in I_{\text{S}}}\sum_{t=1}^{T^{(s)}} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}^{(s)}} & (i \in I_{\text{C}}), \\ \displaystyle\sum_{s\in I_{\text{S}}}\sum_{t=1}^{T^{(s)}} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}^{(s)}} & (i \in I_{\text{O}} \cup I_{\text{V}}). \end{cases}\end{aligned} \tag{B.2}$$

B.1.3 Supplemental Explanation for (2.22): Context Unit Case

$$\begin{aligned}
\frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}^{(s)}} &= \frac{\partial \ln L_{\text{out}}}{\partial c_{t,i}^{(s)}} \frac{\partial c_{t,i}^{(s)}}{\partial u_{t,i}^{(s)}} + \frac{\partial \ln L_{\text{out}}}{\partial u_{t+1,i}^{(s)}} \frac{\partial u_{t+1,i}^{(s)}}{\partial u_{t,i}^{(s)}} \\
&= \frac{\partial \ln L_{\text{out}}}{\partial c_{t,i}^{(s)}} \left\{1 - (c_{t,i}^{(s)})^2\right\} + \frac{\partial \ln L_{\text{out}}}{\partial u_{t+1,i}^{(s)}} \left(1 - \frac{1}{\tau_i}\right) \\
&= \left\{1 - (c_{t,i}^{(s)})^2\right\} \left\{ \sum_{k \in I_{\text{C}}} \frac{\partial \ln L_{\text{out}}}{\partial u_{t+1,k}^{(s)}} \frac{\partial u_{t+1,k}^{(s)}}{\partial c_{t,i}^{(s)}} + \sum_{k \in I_{\text{O}} \cup I_{\text{V}}} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,k}^{(s)}} \frac{\partial u_{t,k}^{(s)}}{\partial c_{t,i}^{(s)}} \right\} \\
&\qquad + \left(1 - \frac{1}{\tau_i}\right) \frac{\partial \ln L_{\text{out}}}{\partial u_{t+1,i}^{(s)}} \\
&= \left\{1 - (c_{t,i}^{(s)})^2\right\} \left\{ \sum_{k \in I_{\text{C}}} \frac{w_{ki}}{\tau_k} \frac{\partial \ln L_{\text{out}}}{\partial u_{t+1,k}^{(s)}} + \sum_{k \in I_{\text{O}} \cup I_{\text{V}}} w_{ki} \frac{\partial \ln L_{\text{out}}}{\partial u_{t,k}^{(s)}} \right\} \\
&\qquad + \left(1 - \frac{1}{\tau_i}\right) \frac{\partial \ln L_{\text{out}}}{\partial u_{t+1,i}^{(s)}} \qquad (i \in I_{\text{C}}).
\end{aligned} \tag{B.3}$$

B.1.4 Supplemental Explanation for (2.22): Output Unit Case

$$\frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}^{(s)}} = \frac{\partial \ln L_{\text{out}}}{\partial y_{t,i}^{(s)}} \frac{\partial y_{t,i}^{(s)}}{\partial u_{t,i}^{(s)}} = \left\{1 - (y_{t,i}^{(s)})^2\right\} \frac{\hat{y}_{t,i}^{(s)} - y_{t,i}^{(s)}}{v_{t,i}^{(s)}} \qquad (i \in I_{\text{O}}). \tag{B.4}$$

B.1.5 Supplemental Explanation for (2.22): Variance Unit Case

$$\begin{aligned}
\frac{\partial \ln L_{\text{out}}}{\partial u_{t,i}^{(s)}} &= \frac{\partial \ln L_{\text{out}}}{\partial v_{t,i}^{(s)}} \frac{\partial v_{t,i}^{(s)}}{\partial u_{t,i}^{(s)}} \\
&= \left(-\frac{1}{2v_{t,i}^{(s)}} + \frac{(y_{t,i}^{(s)} - \hat{y}_{t,i}^{(s)})^2}{2(v_{t,i}^{(s)})^2} \right) v_{t,i}^{(s)} \\
&= -\frac{1}{2} + \frac{(\hat{y}_{t,i}^{(s)} - y_{t,i}^{(s)})^2}{2v_{t,i}^{(s)}} \qquad (i \in I_{\text{V}}).
\end{aligned} \tag{B.5}$$

Appendix C

Predictive Learning of Fluctuating Temporal Sequences

C.1 Learning of Multiple Fluctuating Lissajous Curves

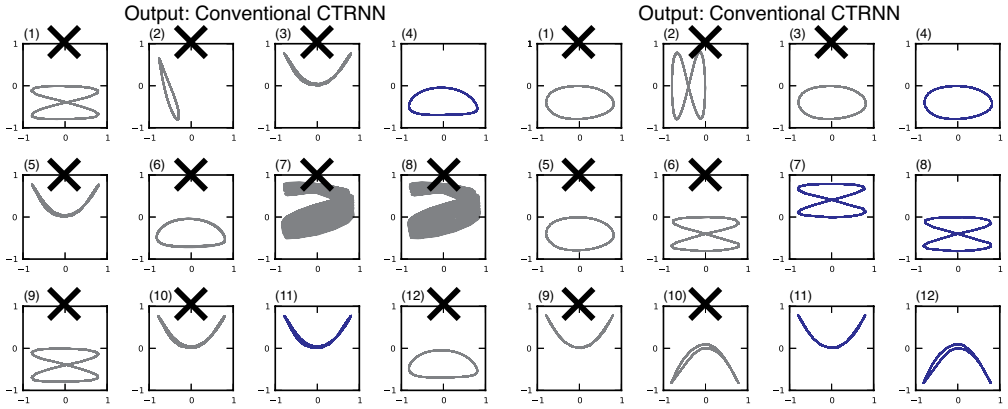
C.1.1 Supplemental Explanation for Target Sequences in (3.2)

The equations for the fluctuating Lissajous curves in the training data in Chapter 3 (Section 3.2) are as follows:

$$(\hat{y}^{(1)}_{t,1}, \hat{y}^{(1)}_{t,2}) = (\hat{Y}_{t,1} + \hat{\epsilon}^{(1)}_{t,1}, \hat{Y}_{t,2} + \hat{\epsilon}^{(1)}_{t,2}), \tag{C.1}$$
$$(\hat{y}^{(2)}_{t,1}, \hat{y}^{(2)}_{t,2}) = (-\hat{Y}_{t,1} + \hat{\epsilon}^{(2)}_{t,1}, \hat{Y}_{t,2} + \hat{\epsilon}^{(2)}_{t,2}), \tag{C.2}$$
$$(\hat{y}^{(3)}_{t,1}, \hat{y}^{(3)}_{t,2}) = (\hat{Y}_{t,2} + \hat{\epsilon}^{(3)}_{t,1}, \hat{Y}_{t,1} + \hat{\epsilon}^{(3)}_{t,2}), \tag{C.3}$$
$$(\hat{y}^{(4)}_{t,1}, \hat{y}^{(4)}_{t,2}) = (\hat{Y}_{t,2} + \hat{\epsilon}^{(4)}_{t,1}, -\hat{Y}_{t,1} + \hat{\epsilon}^{(4)}_{t,2}), \tag{C.4}$$
$$(\hat{y}^{(5)}_{t,1}, \hat{y}^{(5)}_{t,2}) = (\hat{Y}_{t,1} + \hat{\epsilon}^{(5)}_{t,1}, \hat{Y}_{t,4} + \hat{\epsilon}^{(5)}_{t,2}), \tag{C.5}$$
$$(\hat{y}^{(6)}_{t,1}, \hat{y}^{(6)}_{t,2}) = (-\hat{Y}_{t,1} + \hat{\epsilon}^{(6)}_{t,1}, \hat{Y}_{t,4} + \hat{\epsilon}^{(6)}_{t,2}), \tag{C.6}$$
$$(\hat{y}^{(7)}_{t,1}, \hat{y}^{(7)}_{t,2}) = (\hat{Y}_{t,4} + \hat{\epsilon}^{(7)}_{t,1}, \hat{Y}_{t,1} + \hat{\epsilon}^{(7)}_{t,2}), \tag{C.7}$$
$$(\hat{y}^{(8)}_{t,1}, \hat{y}^{(8)}_{t,2}) = (\hat{Y}_{t,4} + \hat{\epsilon}^{(8)}_{t,1}, -\hat{Y}_{t,1} + \hat{\epsilon}^{(8)}_{t,2}), \tag{C.8}$$
$$(\hat{y}^{(9)}_{t,1}, \hat{y}^{(9)}_{t,2}) = (\hat{Y}_{t,3} + \hat{\epsilon}^{(9)}_{t,1}, \hat{Y}_{t,2} + \hat{\epsilon}^{(9)}_{t,2}), \tag{C.9}$$
$$(\hat{y}^{(10)}_{t,1}, \hat{y}^{(10)}_{t,2}) = (-\hat{Y}_{t,3} + \hat{\epsilon}^{(10)}_{t,1}, \hat{Y}_{t,2} + \hat{\epsilon}^{(10)}_{t,2}), \tag{C.10}$$
$$(\hat{y}^{(11)}_{t,1}, \hat{y}^{(11)}_{t,2}) = (\hat{Y}_{t,2} + \hat{\epsilon}^{(11)}_{t,1}, \hat{Y}_{t,3} + \hat{\epsilon}^{(11)}_{t,2}), \tag{C.11}$$
$$(\hat{y}^{(12)}_{t,1}, \hat{y}^{(12)}_{t,2}) = (\hat{Y}_{t,2} + \hat{\epsilon}^{(12)}_{t,1}, -\hat{Y}_{t,3} + \hat{\epsilon}^{(12)}_{t,2}). \tag{C.12}$$

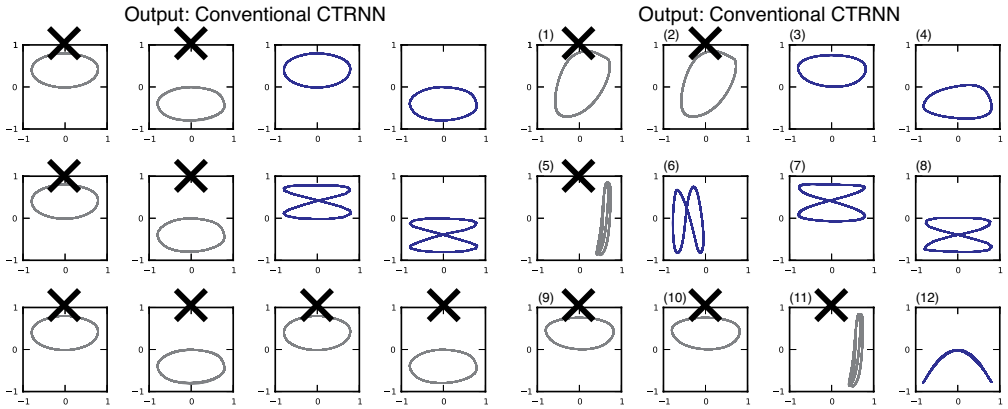
C.1.2 Different Cases of Learning Failure with CTRNNs

Figure C.1 shows the results for other CTRNNs in which different randomly chosen values were used for parameter initialization before predictive learning. For example, in Fig. C.1(b), patterns with smaller noise variance were corrupted by patterns with larger noise variance, in the same manner as in Fig. 3.2. In Fig. C.1(d), untrained attractors appeared (such as (1, 2)), and in Fig. C.1(f), almost none of the training patterns were learned.



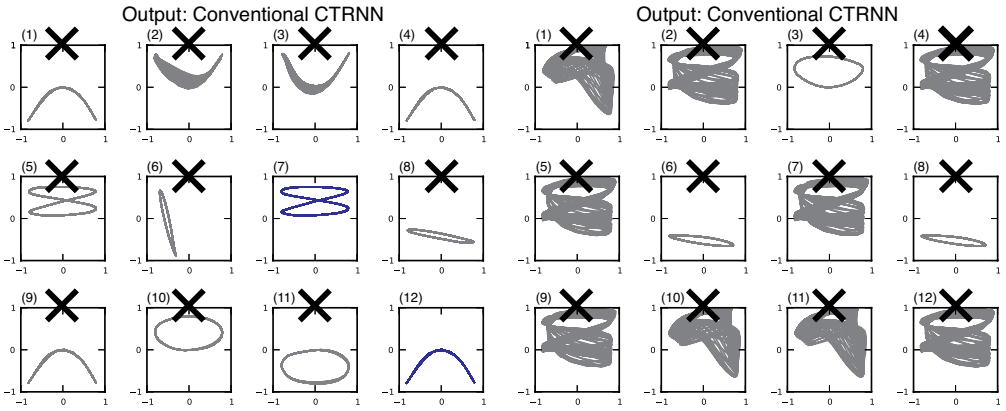
(a) Case 1.

(b) Case 2.

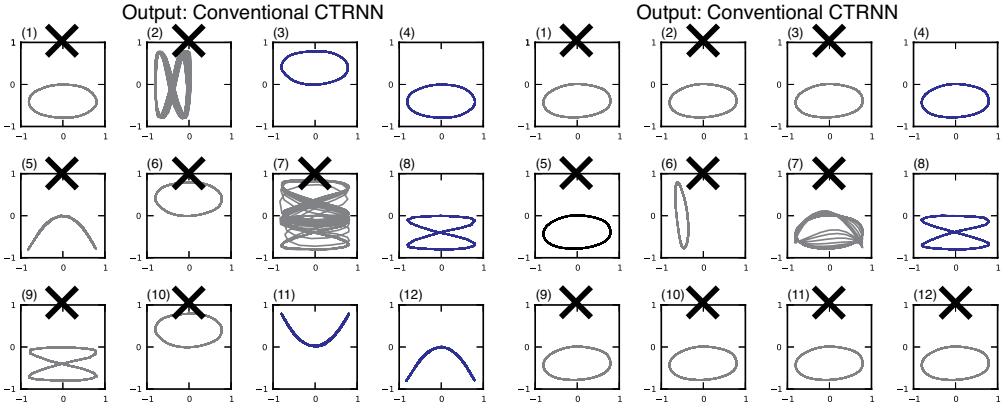


(c) Case 3.

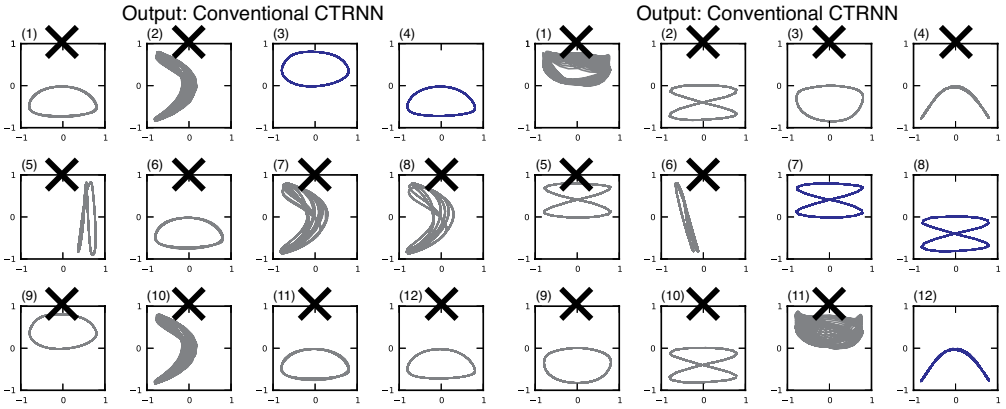
(d) Case 4.



(e) Case 5. (f) Case 6.



(g) Case 7. (h) Case 8.



(i) Case 9. (j) Case 10.

Figure C.1 Phase plots of output states generated by CTRNNs initialized with different random values. The crosses mark cases of failure.

Appendix D

Predictive Learning to Develop Flexible Behavior

D.1 Acceleration of Network Training

To accelerate network training, the adaptive learning rate scheme [44, 71] was employed. In this scheme, the learning rate α is also optimized during the learning process based on the change of a total error before and after updating parameters in the following way:

1. For each learning step n, updated parameters are tentatively computed by (2.17) and (2.19) using the learning rate α (α_{share} or α_{init}), and the total error rate r defined by

$$r = \frac{\sum_{s \in I_{\text{S}}} E^{(s)}(\boldsymbol{\theta}'_{\text{share}}(n), \boldsymbol{\theta}'_{\text{init}}(n))}{\sum_{s \in I_{\text{S}}} E^{(s)}(\boldsymbol{\theta}_{\text{share}}(n), \boldsymbol{\theta}_{\text{init}}(n))}, \tag{D.1}$$

$$E^{(s)}(\boldsymbol{\theta}_{\text{share}}, \boldsymbol{\theta}_{\text{init}}) = \sum_{t=1}^{T^{(s)}} \sum_{i \in I_{\text{O}}} (y_{t,i}^{(s)}(\boldsymbol{\theta}_{\text{share}}, \boldsymbol{\theta}_{\text{init}}) - \hat{y}_{t,i}^{(s)})^2, \tag{D.2}$$

   where $(\boldsymbol{\theta}_{\text{share}}(n), \boldsymbol{\theta}_{\text{init}}(n))$ and $(\boldsymbol{\theta}'_{\text{share}}(n), \boldsymbol{\theta}'_{\text{init}}(n))$ are the current parameters and the tentatively updated parameters, respectively, is also computed.

2. If $r_{\text{th}} < r$, then α is replaced with $\alpha_{\text{dec}}\alpha$, and the procedure returns to step (1) without updating the current parameters $(\boldsymbol{\theta}_{\text{share}}(n), \boldsymbol{\theta}_{\text{init}}(n))$. Otherwise the procedure moves to step (3)

3. If $r < 1$, then α is replaced with $\alpha_{\text{inc}}\alpha$. The current parameters $(\boldsymbol{\theta}_{\text{share}}(n), \boldsymbol{\theta}_{\text{init}}(n))$ are updated with $(\boldsymbol{\theta}'_{\text{share}}(n), \boldsymbol{\theta}'_{\text{init}}(n))$ and the procedure moves to the next learning step $n + 1$.

The present study used $r_{\text{th}} = 1.1$, $\alpha_{\text{dec}} = 0.7$, and $\alpha_{\text{inc}} = 1.05$, which were determined by reference to [44, 71]. The upper limit of 1000 for this iteration process was put at each learning step.

D.2 Reaction Time Measurement

To measure reaction times, numerical sumulations were conducted instead of actual cooperative interactions by reutilizing the recorded 32 pattern visuo-proprioceptive sequences used in the learning phase. During generation phases, the network received sensory input values derived from recorded data for both the two-dimensional object position and the automatically controlled two-dimensional head joint angles, and from its own predictions of four-dimensional arm joint angles. In the experiments for generating forward prediction dynamics, differences between the arm movement state predicted by the trained network and the state recorded in the sequence data at each time step were computed.

Reaction times were measured by counting time steps before the computed differences became sufficiently small. More specifically, the sum of the mean squared errors of predicted four-dimensional arm joint angle state $E_t^{(s)}$ at each time step t for each sequence s was computed as follows:

$$E_t^{(s)} = \frac{1}{2} \sum_{i \in I_{\text{A}}} (y_{t,i}^{(s)} - \hat{y}_{t,i}^{(s)})^2, \tag{D.3}$$

where $I_{\text{A}} \subset I_{\text{O}}$ is the index set for the output units provided for the prediction of arm joint angles. After computing the above values, the time steps from when the object was moved by the other-robot (branching point) until $E_t^{(s)}$ became less than a threshold $E_{\text{th}} = 0.01$ (starting point of cooperative behavior), which was empirically determined, were counted. These computations were conducted for each generation condition, as shown in Figs. 5.7 and 5.11, by using 10 sample networks with differently randomized initial parameters trained with each learning condition. The values shown in the figures are mean values over the 10 trained sample networks, each of which generated 32 sequences including all combinations of the five transitions of the two action primitives (160 branches).

Publications

Journal Articles

1. Shingo Murata, Yuichi Yamashita, Hiroaki Arie, Tetsuya Ogata, Shigeki Sugano, and Jun Tani, "Learning to Perceive the World as Probabilistic or Deterministic via Interaction with Others: A Neuro-Robotics Experiment," *IEEE Transactions on Neural Networks and Learning Systems*, Published Online, pp.1–19, November 2015.

2. Shingo Murata, Hiroaki Arie, Tetsuya Ogata, Shigeki Sugano, and Jun Tani, "Learning to Generate Proactive and Reactive Behavior Using a Dynamic Neural Network Model with Time-Varying Variance Prediction Mechanism," *Advanced Robotics*, Vol. 28, Issue 17, pp. 1189–1203, September 2014.

3. Shingo Murata, Jun Namikawa, Hiroaki Arie, Shigeki Sugano, and Jun Tani, "Learning to Reproduce Fluctuating Time Series by Inferring Their Time-dependent Stochastic Properties: Application in Robot Learning via Tutoring," *IEEE Transactions on Autonomous Mental Development*, Vol. 5, Issue 4, pp. 298–310, December 2013.

International Conferences (Full Papers)

1. Shingo Murata, Saki Tomioka, Ryoichi Nakajo, Tatsuro Yamada, Hiroaki Arie, Tetsuya Ogata, and Shigeki Sugano, "Predictive Learning with Uncertainty Estimation for Modeling Infants' Cognitive Development with Caregivers: A Neurorobotics Experiment," In *Proceedings of the Fifth Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob 2015)*, pp. 302–307, Providence, USA, August 2015.

2. Shingo Murata, Yuichi Yamashita, Hiroaki Arie, Tetsuya Ogata, Jun Tani, and Shigeki Sugano, "Generation of Sensory Reflex Behavior versus Intentional Proactive Behavior in Robot Learning of Cooperative Interactions

with Others," In *Proceedings of the Fourth Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob 2014)*, pp. 242–248, Genoa, Italy, October 2014.

3. Shingo Murata, Hiroaki Arie, Tetsuya Ogata, Jun Tani, and Shigeki Sugano, "Learning and Recognition of Multiple Fluctuating Temporal Patterns Using S-CTRNN," In *Proceedings of the 24th International Conference on Artificial Neural Networks (ICANN 2014)*, pp. 9–16, Hamburg, Germany, September 2014.

4. Shingo Murata, Jun Namikawa, Hiroaki Arie, Jun Tani, and Shigeki Sugano, "Development of Proactive and Reactive Behavior via Meta-Learning of Prediction Error Variance," In *Proceedings of the 20th International Conference on Neural Information Processing (ICONIP 2013)*, pp. 537–544, Daegu, Korea, November 2013.

5. Shingo Murata, Jun Namikawa, Hiroaki Arie, Jun Tani, and Shigeki Sugano, "Learning to Reproduce Fluctuating Behavioral Sequences Using a Dynamic Neural Network Model with Time-Varying Variance Estimation Mechanism," In *Proceedings of the Third Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob 2013)*, pp.1–6, Osaka, Japan, August 2013.

International Conferences (Abstracts)

1. Shingo Murata, Yuichi Yamashita, Hiroaki Arie, Tetsuya Ogata, Jun Tani, and Shigeki Sugano, "Neuro-Dynamical Accounts for Postdiction," *The 19th Annual Meeting of the Association for the Scientific Study of Consciousness (ASSC 19)*, Paris, France, July 2015.

2. Shingo Murata, Yuichi Yamashita, Hiroaki Arie, Tetsuya Ogata, Jun Tani, and Shigeki Sugano, "Self-Organization of Distinct Neural Mechanisms for Adaptive Behavior," *The 2014 IEEE International Conference on Robotics and Automation (ICRA 2014), Neurobiologically Inspired Robotics Workshop: Incorporating Brain Processing into Robots Might for Better Autonomy*, Hong Kong, China, June 2014.

3. Shingo Murata, Yuichi Yamashita, Tetsuya Ogata, Hiroaki Arie, Jun Tani, and Shigeki Sugano, "Altered Prediction of Uncertainty Induced by Network Disequilibrium: A Neuro-Robotics Study," *Computational Psychiatry 2013*, Miami, USA, October 2013.

Domestic Conferences (in Japanese)

1. 村田真悟，山下祐一，有江浩明，尾形哲也，谷淳，菅野重樹: 予測誤差最小化原理に基づくポストディクションの構成論的理解，発達神経科学学会 第4回大会，大阪，2015年9月.

2. 村田真悟，山下祐一，有江浩明，尾形哲也，谷淳，菅野重樹: 異なる神経メカニズムによる能動的・受動的行動の選択，日本機械学会ロボティクス・メカトロニクス講演会2014，P3P2-Q03，富山，2014年5月.

3. 村田真悟，山下祐一，有江浩明，尾形哲也，谷淳，菅野重樹: 予測精度の予測に基づいた能動的・受動的な適応行動の生成学習，人工知能学会全国大会2014，2K4-OS-04a-3，愛媛，2014年5月.

4. 村田真悟，有江浩明，尾形哲也，谷淳，菅野重樹: S-CTRNNを用いた複数時系列パターンの記憶学習，情報処理学会 第76回全国大会，3C-6，東京，2014年3月.

5. 村田真悟，並川淳，有江浩明，谷淳，菅野重樹: 再帰結合神経回路モデルによるばらつきを伴った運動軌道の確率的構造の獲得，第31回日本ロボット学会学術講演会予稿集，Vol. 31st，2C2-02，東京，2013年9月.

6. 村田真悟，並川淳，有江浩明，谷淳，菅野重樹: プロアクティブ・リアクティブな行為とその自律的な切り替えの学習，日本ロボット学会第30回記念学術講演会予稿集，Vol. 30th，N3-5，北海道，2012年9月.

7. 村田真悟，有江浩明，谷淳，菅野重樹: 自らの行為経験に基づいた言語学習モデル 複数の文法構造をもつ文の学習における汎化，日本機械学会ロボティクス・メカトロニクス講演会2011，2P2-M03，岡山，2011年5月.

Other International Conferences (Full Papers)

1. Tatsuro Yamada, Shingo Murata, Hiroaki Arie, and Tetsuya Ogata, "Attractor Representations of Language—behavior Structure in a Recurrent Neural Network for Human—robot Interaction," In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015)*, Hamburg, Germany, September 2015.

2. Ryoichi Nakajo, Shingo Murata, Hiroaki Arie, and Tetsuya Ogata, "Acquisition of Viewpoint Representation in Imitative Learning from Own Sensory-Motor Experiences," In *Proceedings of the Fifth Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob 2015)*, pp. 326–331, Providence, USA, August 2015.

3. Kuniyuki Takahashi, Tetsuya Ogata, Hadi Tjandra, Shingo Murata, Hiroaki Arie, and Shigeki Sugano, "Tool-body Assimilation Model based on Body Babbling and a Neuro-dynamical System for Motion Generation," In *Proceedings of the 24th International Conference on Artificial Neural Networks (ICANN 2014)*, pp. 363–370, Hamburg, Germany, September 2014.

Other Domestic Conferences (in Japanese)

1. 平野加依，村田真悟，張耀宇，有江浩明，尾形哲也: 神経回路モデルを用いた人間とロボットの模倣インタラクション解析，第 16 回計測自動制御学会システムインテグレーション部門講演会，愛知，2015 年 12 月.

2. 山田竜郎，村田真悟，有江浩明，尾形哲也: インタラクションを行うロボットの神経回路上アトラクタにおける言語と行動の動的統合，第 33 回日本ロボット学会学術講演会予稿集，Vol. 33rd，1B3-05，東京，2015 年 9 月.

3. 冨岡咲希，村田真悟，中條亨一，山田竜郎，有江浩明，尾形哲也，菅野重樹：養育者—幼児間インタラクションの認知ロボティクスモデル 予測学習とその不確実性に基づく注意対象の遷移，日本赤ちゃん学会第 15 回学術集会，香川，2015 年 6 月.

4. 中條亨一，村田真悟，有江浩明，尾形哲也: 再帰型神経回路モデルを用いた観察視点の獲得によるロボットの模倣学習，人工知能学会全国大会 2015，2D1-OS-12a-2，北海道，2015 年 5 月.

5. 鈴木彼方，高橋城志，Hadi Tjandra，村田真悟，菅野重樹，尾形哲也: 再帰神経回路モデルによる分散予測を用いた柔軟関節ロボットの身体ダイナミクスの探索，日本機械学会ロボティクス・メカトロニクス講演会 2015，2P1-S06，京都，2015 年 5 月.

6. 山田竜郎，村田真悟，有江浩明，尾形哲也: 人間ロボットインタラクションを目的とした神経回路による言語と行動のアトラクタ表現，情報処理学会第 77 回全国大会，5T-01，京都，2015 年 3 月.

7. 高橋城志，尾形哲也，Hadi Tjandra，野田邦昭，村田真悟，有江浩明，菅野重樹: 神経回路モデルと身体バブリングによる道具身体化と道具機能の獲得，日本機械学会ロボティクス・メカトロニクス講演会 2014，P3P2-P02，富山，2014 年 5 月．

8. 高橋城志，尾形哲也，Hadi Tjandra，野田邦昭，村田真悟，有江浩明，菅野重樹: 身体バブリングと再帰結合型神経回路モデルによる道具身体化〜深層学習による画像特徴量抽出〜，人工知能学会全国大会 2014，1I4-OS-09a-4，愛媛，2014 年 5 月．

9. Hadi Tjandra，高橋城志，村田真悟，有江浩明，山口雄紀，尾形哲也，菅野重樹: 神経力学モデルと身体バブリングに基づく道具身体化と動作生成，情報処理学会 第 76 回全国大会，1S-4，東京，2014 年 3 月．