Learning of Language Grounding in Robot Behavior by Recurrent Neural Networks

リカレントニューラルネットワークによる言語と行動のグラウンディングの学習

February 2019

Tatsuro YAMADA
山田竜郎

Learning of Language Grounding in Robot Behavior by Recurrent Neural Networks

リカレントニューラルネットワークによる言語と行動のグラウンディングの学習

February 2019

Waseda University,
Graduate School of Fundamental Science and Engineering,
Department of Intermedia Studies,
Research on Intelligence Dynamics and Representation Systems

Tatsuro YAMADA
山田竜郎

Abstract

Natural language is the most powerful tool for expressing our requests to other agents. Therefore, service robots must be able to understand natural language to flexibly work by responding to human requirements or to effectively work together with humans. Here, it is not enough for robots to communicate with us in a purely verbal manner (e.g., chat or question answering). Robots in the real world must work with their own bodies to have some effects on the environment by understanding humans' instructions, while also being able to explain current situations, occurring events, or their own behavior using linguistic expressions. In other words, they must have the capability to use language in a form that is grounded in the real world. To arbitrarily design mapping between language, which is a discrete system, and the referents in the real world, which is a continuous and dynamical system, for intelligent machines is notoriously difficult, stated as the symbol grounding problem.

The objective of this thesis is to build a computational framework that attains the following two capabilities related to the language grounding: (a) generation of robot behavioral sequences in response to linguistic instructions and (b) generation of linguistic descriptions given robot behavioral sequences. The first capability is clearly a requirement of service robots. The second capability is also required in order to interpret service robot behavior. Because robots make physical interaction with humans, there is a risk of injury. To easily investigate the cause of accidents when they occur, robotic system behavior should be easily understandable.

To ground linguistic expressions in robot behavior, we focus on three important issues: (1) solving the context dependency of the meanings of linguistic

expressions, (2) understanding both content words (e.g., nouns, verbs, adjectives, and adverbs) and function words (e.g., conjunctions and prepositions), and (3) bidirectional conversion between language and behavior.

First, the meanings of linguistic expressions depend on the current context that an agent is situated. For instance, to respond to the instruction "grasp the red ball," a robot is required to generate different trajectories of joint angles in accordance with the position of the red ball. Thus, the robot must solve such context dependency of the meaning of the language instruction by observing the current context.

Secondly, linguistic expressions consist of not only content words but also function words. In general, the content words have their own meaning grounded in the world. In contrast, function words have no explicit referents or clearly grounded meanings; they contribute to the meaning of a phrase or a sentence by being combined with content words together. Humans naturally build sentences that include both content and function words. Therefore, intelligent systems must understand expressions consisting of both.

The third requirement is to build a system that can convert language and behavior bidirectionally. The bidirectionality or symmetric bias on inference is a distinctive characteristic of humans' usage of symbols. Because we communicate with each other on the premise of this bidirectionality, it is also a requirement for robots to have such a bidirectional conversion capability to achieve efficient interaction.

To address the aforementioned issues, we propose a machine learning framework that employs a recurrent neural network (RNN). An RNN is a type of neural network extended to handle sequential or temporal data. RNNs learn a mapping from one sequence to another through its context feature space. Our approaches are summarized as follows: (1) bottom-up learning of conversion between language and behavior by an RNN-based encoder–decoder model (EDM), (2) integration of multimodal information, and (3) representation sharing between language and robot behavior.

First, we use an RNN-based EDM, which is a learning framework originally proposed in the field of natural language processing for applications such as ma-

chine translation and question answering. An encoder RNN projects a source sentence into a fixed-dimensional vector. Then, a decoder RNN produces a sentence in the target domain by decoding the representation. The current study applies this framework to convert between language and behavior. Because the relations between sequences are totally learned from data, we do not have to manually design rules and knowledge for language grounding.

As the second approach, our model integrates multimodal information together. By this extension, we address the first issue: the context dependent relations between language and behavior. The model learns to generate appropriate behavior by integrating a word sequence with visual and proprioceptive information. Moreover, the second issue of understanding both content words and function words is also addressed. The encoder RNN embeds a word sequence into a fixed-dimensional representation in accordance with its meaning. Here, content words are embedded in a form that is integrated with the current visual and proprioceptive context. On the other hand, function words are embedded in a way integrated with other content words and reflecting their own function as logical operators.

As the third approach, we propose an extension of the framework to a two-coupled EDM architecture in order to accomplish the bidirectional conversion between language and behavior. More precisely, we use two recurrent autoencoder (RAE)-type EDMs. One EDM learns to embed behavioral sequences as fixed-dimensional representations in a way that allows the sequences to be regenerated from the representations. The other EDM embeds linguistic sequences in a similar manner. Here, we introduce a loss function whereby the representations of a behavior and its corresponding description approach each other in the latent vector space. This loss allows the two EDMs' internal representation spaces to be shared. Through this shared representation, robot behaviors and linguistic expressions are bidirectionally converted, conditioned on visual context.

We evaluate these proposed approaches through the following three robot experiments: (1) unidirectional conversion from sentences including only content words to robot behavior, (2) unidirectional conversion from sentences including both content and function words to robot behavior, and (3) bidirectional conver-

sion between language and robot behavior. These three evaluation experiments correspond to the aforementioned three issues.

This thesis is organized into seven chapters. Chapter 1 provides the background, the research objective, and an overview of our approaches as an introduction of the current study.

Chapter 2 reviews existing studies in related fields. First, we review the artificial intelligence research in the earliest days, where intelligence was considered to be a purely symbolic process and rule-based systems were designed on the basis of such a philosophy. After referring to the difficulties researchers have been faced with, we review learning-based approaches. Specifically, we introduce existing works in the field of cognitive developmental robotics. Then, we review the recent trend in grounding learning, namely deep learning methods. Finally, we provide a brief survey of these existing works, indicate untouched issues, and clarify our study's position on them.

In Chapter 3, to address the first two issues, we propose an RNN-based EDM extended by multimodal input. It learns to unidirectionally convert language (resp., behavior) into behavior (resp., language). It first encodes a linguistic instruction (resp., behavioral sequence) into a fixed dimensional space and then decodes the latent representation as a corresponding behavioral sequence (resp., linguistic description). By encoding multimodal information together, namely vision and proprioception, our model solves the context dependency of the meanings of sentences.

In Chapter 4, we perform a learning experiment using a humanoid robot to evaluate the proposed framework on the first requirement — the context dependency of language. The task is the unidirectional conversion from linguistic instructions including only content words into robot behavioral sequences. The proposed model successfully learns to convert three- or four-word instructions into robot behavior by encoding visual information together. Moreover, we investigate how the model internally represents the grounding relations. This analysis shows that the links of instructions and behaviors are embedded in the middle of the RNN's attractor dynamics.

Chapter 5 evaluates the proposed framework in terms of the second require-

ment — understanding of both content and function words. We add some logic words, such as "not", "and", and "or" to instruction sentences as examples of function words, and the model successfully learns the task. The analysis shows that the content words are encoded in a form integrated together with the current visual and proprioceptive information. The logic words are simultaneously represented together with content words by the model in accordance with their own function as a logical operator.

In Chapter 6, we extend the proposed framework in two-coupled EDM architecture to achieve the third requirement — bidirectional conversion between language and behavior. The proposed system consists of two different EDMs — one for behavior and the other for language — and trains them so that their internal representation spaces are shared. This binding learning enables the model to convert language and behavior bidirectionally, conditioned on visual context. We evaluate this extended framework in a robot experiment. After learning, the model successfully produces a description given a robot behavioral sequence through the shared representation. The model can also execute an appropriate behavior in response to a language command, conditioned on the visual context. Analysis of the latent representation spaces shows that the behavioral sequences are embedded in a semantic manner by being learned jointly with linguistic descriptions.

Chapter 7 concludes this thesis by summarizing the accomplishments of our study in terms of language grounding in robot behavior and reviewing remaining issues and future research directions.

Acknowledgements

This work was carried out at the Graduate School of Fundamental Science and Engineering at Waseda University in 2016–2019. I thank the institute for providing me with excellent research facilities. Here, I would like to express my sincere appreciation to those who were involved in my study and life in the past three years.

Firstly, I would like to gratefully thank my supervisor Prof. Tetsuya Ogata for willingly allowing me to belong to his laboratory from my master's course. Over the last five years, he have provided me with attentive guidance and significantly important comments based on his transcendent insight and extensive experience. He taught me all about research, including how to think, how to perform, how to tell, and how to live.

I would like to express my deep gratitude to Prof. Yasuhiro Oikawa and Prof. Takashi Kawai of the Department of Intermedia Studies, Waseda University for giving me meticulous comments and constructive suggestions through the review of this thesis. I also owe a very important debt to Dr. Tetsuji Ogawa of the Department of Communications and Computer Engineering (Waseda University), Dr. Shingo Shimoda of the Institute of Physical and Chemical Research (RIKEN), and Prof. Angelo Cangelosi of the University of Manchester for their insightful comments. Prof. Cangelosi very kindly accepted me as a visiting student in the University of Plymouth from June 2017 to November 2017 and provided me with penetrating advice and invaluable support that greatly contributed to my study.

I would like to show my deep appreciation to Dr. Hiroaki Arie, Dr. Shingo Murata, Dr. Yuki Suga, Dr. Hiroki Mori, Dr. Kuniaki Noda, Dr. Kuniyuki Takahashi, and Dr. Kazuma Sasaki. Their skillful comments extraordinarily

helped me learn a scientific perspective and engineering methodology. Without their great contribution, this work would not have been possible. I would also like to thank the laboratory's secretaries, Mrs, Naomi Nakata and Ms. Yumiko Nakano for enormously supporting me to conduct my research.

I would like to offer my special thanks to the members of "Language Robotics" group of Ogata Lab., Mr. Hiroyuki Matsunaga, Ms. Saki Ito, and Mr. Satoshi Murasawa. Their extremely fascinating ideas that I would hardly come up with alone greatly inspired me to refine and boost my thought. I would also like to express other members in Ogata Lab., students in Sugano Lab. belonging to the joint research group, and people in Centre for Robotics and Neural Systems in the University of Plymouth for taking exciting discussion on research and comfortable chats in everyday coffee breaks and lunch.

My heartfelt appreciation goes to Prof. Taikan Yamada, my supervisor from 2012 to 2014. He generously gave me abundant knowledge about "symbol" and "language" and brought out my interest in them. I was taught the importance of reading literature and pleasure of thinking about philosophical issues by him. His suggestive guidance definitely laid the foundation of this work.

From 2016 to 2017, I received financial support from the program for leading graduate schools, "Graduate Program for Embodiment Informatics" of the Ministry of Education, Culture, Sports, Science, and Technology. This program also provided me with great opportunity to advance my study, such as lectures, workshops, and an internship. The discussion with people in the program brought me interdisciplinary perspective on my research target. This work was also supported by a Japan Society for the Promotion of Science Grant-in-Aid for JSPS Research Fellows (No. 17J10580).

Finally, special thanks to my family and friends. Without their unconditional support and encouragement, I would never finished this study. Especially, I would like to express the deepest appreciation to my parents who have tolerantly given moral support to me. I can never repay them for all their love deeper than the sea.

Tokyo, 8th, February, 2019 Tatsuro Yamada

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Research Objective

Natural language is the most powerful tool for expressing our requests to other agents. Therefore, service robots must be able to understand natural language to flexibly work by responding to human requirements or to effectively work together with humans. Here, it is not enough for robots to be able to communicate with humans in a purely verbal manner; for instance, to engage in enjoyable chat or to answer our queries by using knowledge stored in some database. Robots in the real world must work with their own bodies to have some effects on the environment by understanding humans' instructions, while also being able to explain current situations, occurring events, or their own behavior using linguistic expressions to convey helpful information to humans. In other words, they must have the capability to use language in a form that is *"grounded"* or *"embodied"* in their own sensorimotor experience in the real world. To arbitrarily design mapping between language, which is a discrete system, and the referents in the real world, which is a continuous and dynamical system, for intelligent machines is notoriously difficult, stated as the symbol grounding problem [1].

Given such a background, the objective of this study is to build a computational framework that attains language use grounded in the real world. Specifically, this study focuses on two capabilities that require language grounding: (a) the capability to generate robot behavioral sequences in response to linguistic instructions and (b) the capability to generate linguistic descriptions of robot be-

havioral sequences. The first capability — to behave appropriately in response to human instructions — is clearly required for service robots. Language is the most powerful tool to convey our requests to other agents. We can express a vast range of our own wants by flexibly constructing novel phrases or sentences. The second capability — to describe the robot's own behavior — is also required for interpretability and explainability of robot behavior. Because robots have the ability to physically interact with humans, there is a risk of injury. To easily investigate causes of accidents when they occur, we should design robotic systems in a way that makes their behavior intelligible[1].

To make a system that uses language in a manner grounded in the real world, one of the most fundamental problems is **the context dependency of meanings of linguistic expressions (Requirement 1).** The same sentence can refer very different things depending on the current context. As an example, imagine a simple instruction "pass me the salt" in our daily lives. To achieve this instruction, a robot has to generate much different trajectories of joint angles in accordance with the position of the salt shaker. In contrast, even if the joint angle trajectory is perfectly the same, the linguistic description could be different depending on the context (e.g. object properties, robot's intent). Any grounding methods for robots have to solve such many-to-many relations between language and behavior caused by context dependency of meanings of language by observing the current context.

Here, if we could write down all the possible situations in a specific environment and make a set of finite rigid rules that assign a linguistic expression to each of these situations in advance, there would be no problem. In fact, some robots situated in a factory can work very well under this strategy because workspaces in factories are well structured and controlled environments. It is possible to enumerate most situations that can occur, although the system would still have a possibility of encountering exceptional cases. However, unlike most situations in industrial factories, our living environment is highly changeable and open-ended; new situations almost always differ from the previous ones. It is almost impossible

[1]Although more generally the capability to describe details such as a partner's behavior or the present status of the external environment may be also required, this study does not focus on this. In fact, there have already been studies that focus on these requirements [2, 3].

to make explicit rules that can handle all possible situations in advance.

There have also been many studies that have attempted to build robots that learn the grounding relationships from their own experiences given small or no a-priori knowledge [4, 5]. In fact, humans can learn the relationships between real world matters and their linguistic expressions from our finite number of personal experiences in a form that can generalize to unexperienced situations. In the field of cognitive developmental robotics [4, 6], researchers have modeled humans on the basis of the neuroscientific, psychological, or morphological knowledge and implemented the models in robots to make them learn from their own experiences. Again, they emphasize the importance of "*embodiment*." The concept of "*embodied cognition*" argues that intelligence, which includes the use of language, emerges only from dynamical interaction between an agent's internal cognitive system and their external environment mediated by the sensorimotor organization specific to the agent's body [7]. With this philosophy, experiments have been performed in which robots interact with the physical world and other agents to develop their own embodied intelligent skills [8].

In this study, because we consider situations in which instructions are given in the form of natural language, we have to build **a system that is able to understand expressions that consist of both content words and function words (Requirement 2).** When we consider the basic structure of natural language expressions, in most situations, instructions to robots and descriptions of robot behavior are expressed as phrases or sentences, which consist of multiple words. In a phrase/sentence, each word itself is grounded in some matter in the world. For example, nouns correspond to some entity or category that an entity belongs to. Adjectives refer to the characteristics of entities. Verbs express some action or its effect. By composing these grounded words, the context-dependent meaning of a sentence is constructed. These words are called content words. On the other hand, there are words that have no explicit referents or clearly grounded meanings; these are called function words, and examples include conjunctions and prepositions. Function words are more related to syntactic structure. Thus, they contribute to constructing the meaning of a phrase or a sentence by being combined with content words.

Existing studies aimed at addressing symbol grounding in robots have mainly focused on content words [9, 10, 11]. However, humans always build sentences that include both content and function words unconsciously. In fact, the function words are very important elements because they are strongly involved in the compositionality of language. In the field of formal semantics, the principle of compositionality (also referred to as Frege's principle) models language systems as follows: the meaning of a phrase or a sentence is given as a function of the meanings of its parts (e.g., words) [12]. We can express an infinite number of events that occur in this continuous world by compositionally combining a finite number of words. Specifically, function words extend the linguistically explainable space dramatically. Therefore, we must build a system that deals with both content and function words[2].

The other requirement that robots must satisfy to achieve language grounding is **the capability to convert language and their behavior bidirectionally (Requirement 3).** The bidirectionality or symmetric bias on inference is a distinctive characteristic of humans' usage of symbols, which other primates do not have [13, 14]. Once we humans learn that an apple has the name *"apple,"* we also obtain the knowledge that the word *"apple"* refers to apples. Because humans communicate with each other on the premise of this bidirectionality, it is also required that robots have the same bidirectional conversion capability to achieve efficient interaction. In recent times, many deep learning models that learn to map linguistic instructions into agent action sequences (e.g., in 2D-grid worlds and simulated 3D environments) have been proposed [15, 16, 17, 18, 19]. However, there are very few existing studies that have addressed bidirectional mapping. On the other hand, some studies have applied deep neural networks (NNs) to cross-modal retrieval tasks [20, 21, 22]. In these studies, representations of paired data in different modalities (video, sound, and text) are bound with each other

[2]In fact, words can be further divided into characters. Moreover, in the case of speech, a sequence of characters can also be represented as phonemes, which are finally generated as audio waveforms. If we wish to deal with humans' developmental process, emotion recognition from verbal utterances, or continuously changing variants of Internet slang words (e.g., "amazing", "amaazing", and "amaaaazing"), we may have to consider these kinds of lower-level structures simultaneously. However, this study tackles the grounding problem at the word- and sentence-levels, which in practice is the main part of semantic grounding. Although morphological division is also important, we will consider it in future work.

to organize semantically similar representation spaces among modalities. Given a sample in one modality, data in the other modalities are retrieved through the shared representation space. However, these models can only be utilized for retrieval tasks, and thus are not be able to produce novel samples.

Taken together, we focus on the following three issues that are essentially necessary to achieve grounded language use:

- **Requirement 1 (R1):** Solving context dependency of meanings of linguistic expressions.
- **Requirement 2 (R2):** Understanding both content words and function words.
- **Requirement 3 (R3):** Bidirectional conversion between language and behavior.

1.2 Overview of Our Approaches

To address our three requirements, we propose a machine learning framework that employs a recurrent neural network (RNN). An RNN is a type of NN extended to handle time-series data (and other sequential data) [23, 24]. RNNs can learn to encode variable lengths of sequential data into fixed-dimensional feature vectors, to decode feature vectors as sequential data, and to map a sequence to another sequence through the feature space. This study uses RNN-based models to handle the conversion between linguistic expressions and robot behavioral sequences. Our approaches are summarized as follows:

- **Approach 1 (A1)**: Bottom-up learning of conversion between behavioral sequences and linguistic expressions by an RNN-based encoder–decoder model (EDM).

  Solution: Acquisition of relations between language and behavior without a-priori knowledge or arbitrary rules in an end-to-end manner.

- **Approach 2 (A2)**: Integration of multimodal information.

Solution: Understanding of context-dependent linguistic expressions that include both content and function words.

- **Approach 3 (A3)**: Representation sharing between robot behavior and linguistic expressions.

  Solution: Bidirectional conversion ability.

First, we use RNN-based EDM as a basic framework to convert between linguistic expressions and robot behavioral sequences. The EDM, which is also called a sequence to sequence learning model, is a learning model originally proposed in the field of natural language processing for applications, such as machine translation [25, 26, 27] and question answering [28], and has overwhelmingly outperformed conventional methods, such as rule-based [29] and phrase-based methods [30]. Designing rules for translation manually is much difficult and time-consuming. In contrast, the EDMs can learn the relationships between source domain and target domain in an end-to-end manner from parallel corpus. In their working, an encoder RNN non-linearly projects a source sentence into a fixed-dimensional vector, and then a decoder RNN produces a target sentence by decoding the vector representation. The current study applies this framework to conversion between language and behavior. In a similar to that of machine translation, since the relationships between sequences are learned entirely from the dataset, we do not have to design explicit a-priori knowledge.

As the second approach, our model integrates multimodal information. Through this extension, we address R1. The model learns to generate appropriate behavior conditioned on not only word sequences but also visual and proprioceptive input. Moreover, by combining these two approaches, R2 is also addressed. The encoder of the EDM projects a variable-length word sequence onto a fixed-dimensional space in accordance with the meaning. Here, content words can be embedded in a form that is integrated together with multimodal input. Function words can be simultaneously embedded in a manner integrated with other content words. For example, *"don't"*, which expresses negation, works as a non-linear operation of the internal state of the RNN and then makes the RNN generate opposite behavior. An actual example is demonstrated in Chapter 5.

As the third approach, we propose to extend the framework to the two-coupled EDM architecture — one for behavior and the other for language — and train these EDMs so that their representation spaces are shared. This binding learning allows the model to convert language and behavior bidirectionally. More precisely, we use two coupled recurrent autoencoder (RAE)-type EDMs. One EDM learns to embed behavioral sequences into fixed-dimensional latent representations in a way that allows the original sequences to be regenerated from the representations by the decoder. The other EDM embeds linguistic sequences in a similar manner. Here, we introduce a loss function whereby the representations of a behavior and its description become close in the latent space. Thanks to this loss function, the representation spaces of behavior and language are gradually shared in the learning process. Through the shared representation space, robot behavioral sequences and linguistic descriptions can be bidirectionally converted, conditioned on visual context.

We evaluate the conversion capability of our proposed framework through the following three robot experiments.

- **Evaluation 1**: Unidirectional conversion from language to robot behavior (only content words)
- **Evaluation 2**: Unidirectional conversion from language to robot behavior (both content and function words)
- **Evaluation 3**: Bidirectional conversion between language and robot behavior

These three evaluation experiments correspond to the aforementioned R1, R2, and R3, respectively. The first experiment evaluates the framework on the issue of context dependency of language. We build a learning system that combines our first and second approaches, namely an EDM extended with multimodal input, and we evaluate it with a unidirectional mapping task from word sequences to robot behavioral sequences. This experiment includes three- and four-word sentences that consist of only content words. The behavior to be generated depends on visual context. The model achieves the learning of the semantic relationships

among language, vision, and behavior by experiencing only one-third of the possible situations. Hence, it succeeds in generating appropriate behavior even in unexperienced situations by generalizing the acquired knowledge.

In the second evaluation experiment, although we design a unidirectional task from language to behavior again, the task includes not only content words but also logic words such as "and", "or", and "not" as examples of function words. The model succeeds in appropriately learning a given task. The analysis of internal representations reveals that content words are grounded in the visual and proprioceptive contexts and that the logic words are encoded together with other content words in accordance with their functions as logical operators. For example, the word "not" works as a non-linear transformation to embed orthogonal phrases into the same area in the latent representation space. "Or," which requires behavior execution that looks apparently random, is represented as unstable areas of the RNN's dynamical system.

The third experiment evaluates the proposed method on the third requirement, the bidirectional conversion between language and behavior. We extend the proposed EDM to the two-coupled form. The task imposed on the model is to learn to convert linguistic instructions (descriptions) and robot behavioral sequences bidirectionally conditioned on visual context from parallel data. After training, through the shared latent space, the model successfully generates a description sentence given a robot behavioral sequence. The model also succeeds in generating an appropriate behavior in response to an instruction, depending on the current visual context. Visualization of the latent representations reveals that by being learned jointly with linguistic descriptions, the robot behavioral sequences are encoded in a semantically compositional manner.

1.3 Thesis Organization

This thesis is organized into seven chapters as described in Figure 1.1. Chapter 2 reviews existing studies in related fields and clarifies the position of the current study among them. First, we review the background of artificial intelligence in its earliest days. Intelligence was thought of as a purely symbolic process and

computational systems were designed on the basis of such a philosophy. After referring to the difficulties researchers have been faced with, we review learning-based approaches. Machine learning approaches extract the relationships between language and other modalities from data, and therefore they have no need to arbitrarily design grounding rules. Specifically, we introduce existing works in the field of cognitive developmental robotics, which attempt to understand human cognitive skills in a constructive manner by performing embodied robot experiments. Next, we review the recent trend in grounding learning, namely the deep learning method. Finally, we clarify our study's position on these existing studies.

In Chapter 3, we propose our method to address the first two issues, namely R1 and R2. The proposed learning model is an RNN-based EDM extended by multimodal input, which corresponds to our first and second approaches. It learns to unidirectionally convert language (resp., behavior) into behavior (resp., language). It first encodes a linguistic instruction (resp., behavioral sequence) into a fixed dimensional space and then decodes the latent representation as a corresponding behavioral sequence (resp., linguistic description). By integrating multimodal information together, our model solves the context dependency of the meanings of sentences.

In Chapter 4, we perform the first evaluation experiments using a real humanoid robot. For this experiment, which focuses on R1, we design a unidirectional conversion task from linguistic instructions into robot behavioral sequences. The task includes only content words. The proposed RNN successfully learns to convert linguistic instructions that include at least a verb, an object, and an adverb into robot behavior by encoding visual information together. Moreover, we analyze how the trained RNN represents the grounding relationships as its internal states. The analysis shows that the link between instructions and behavioral sequences is embedded in the middle of the RNN's attractor dynamics.

Chapter 5 evaluates our proposed model on R2. We design a unidirectional task from language into robot behavior again, but the task includes both content words and function words. More precisely, we add some logic words, such as "not", "and", and "or". The model succeeds in learning the given task. The analysis shows that the grounded content words are encoded in a form integrated together

with the visual information and robot's own joint states. The logic words are represented together with other content words by the model in accordance with their functions as logical operators.

In Chapter 6, we extend the framework to the two-coupled EDM architecture for R3. The proposed system consists of two different EDMs — one for behavior and the other for language — and train them so that their internal representation spaces are shared with each other. This binding learning enables the model to convert language and behavior bidirectionally, conditioned on visual input. We evaluate the method on a robot experiment. After training, through the shared latent representation space, the model successfully produces a description sentence given a behavioral sequence. It also succeeds in generating an appropriate behavior by receiving a linguistic command, conditioned on the visual context. Visualization of the representation spaces reveals that by being learned jointly with linguistic descriptions, the robot behavioral sequences are encoded in a semantically compositional manner.

In Chapter 7, we conclude this thesis by summarizing the accomplishments of our study in terms of language grounding in robot behavior and reviewing remaining issues and future research directions.

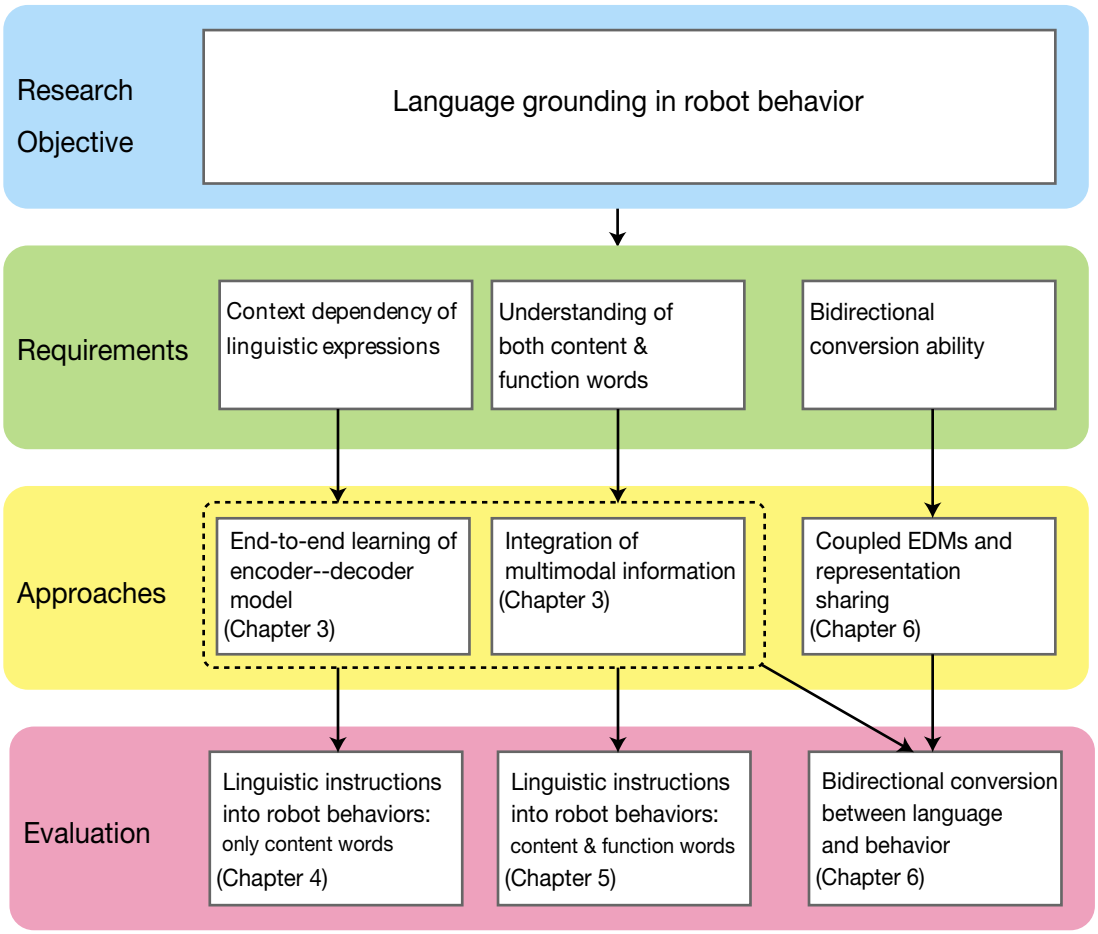

Figure 1.1. Thesis organization.

Chapter 2

Literature Review

2.1 Old-fasioned Artificial Intelligence and Symbol Grounding Problem

At the dawn of "artificial intelligence" in the 1950s, intelligence was considered as a solely symbolic process. Advocated as "physical symbol systems" by Newel and Simon [31, 32], researchers at the earliest stage hypothesized that any kinds of cognitive process can be materialized as manipulations of symbols, which are themselves physical entities and substitute for their referents. This hypothesis seems to have been strongly inspired by the behavior of computers. Given a problem to solve, once a procedure to solve it is modeled as an algorithm and implement it as a program code, the solution can be obtained as the computation ends. It was believed that our intelligence is also such a computational process of physical symbols in a brain and that it can be replicated by modeling the process on another physical substrate. In fact, various kinds of systems were developed based on this idea. These systems calculate an answer in response to a query by using manually designed rules (inference engine) and knowledge base. Examples include ELIZA [33], which is a dialog system driven by simple pattern matching rules, and SHRDLU [34], which manipulates different-shaped/-colored blocks in response to human instructions in a virtual environment. Moreover, beyond the earliest programs that could solve relatively simple tasks (e.g., toy problems) only, studies in the 1980s could provide more powerful tools by focusing on specific

tasks and massively extending the number of rules and size of knowledge that are given by professionals. They were called expert systems and applied to diagnosis, financial system, manufacturing industry, etc. [35]. For another example, Deep Blue [36], which is the first computer that defeated a human professional player in chess, also had many evaluation functions designed by humans to evaluate board states and utilized them to search for better moves.

However, this hypothesis and design methodology have a serious problem. It assumed that the target can be perfectly replaced and manipulated as symbolic representations without any discrepancies from the original or elimination of requisite information. In general, this assumption is not possible when we are situated in the real world. Symbolization always run the risk of creating a gap between the symbol system and its referent. In the case of the chess field and SHRDLU's virtual environment, the referential world is closed and finite, so all the states can be described accurately. However, when we deal with the real world, it is inevitable for gaps to be created. We cannot establish mapping between a discrete symbolic system and the continuous open-ended real world in a one-to-one way. Therefore, symbol systems only work as approximations. This is involved in the so called symbol grounding problem [1]. The symbol grounding problem is explained as follows: how can the symbols that intelligent systems internally possess and manipulate be grounded in the real world? This problem was also pointed out by Searle [37] in his popular thought experiment "Chinese room." In fact, the symbols in such systems that are arbitrarily designed by humans are intrinsically not grounded in real-world matters. The real-world environment in which service robots work is highly changeable in terms of positions of objects and obstacles, light condition, robot's own state, etc. If we attempted to consider all of them, the number of required rules would be infinitely large. It is almost impossible to make explicit rules that can handle all possible situations in advance.

2.2 Cognitive Robotics and Learning-based Approach

In the previous section, we reviewed that arbitrarily designing rules that define relations between symbols and referents in the real world in a top-down manner is too difficult and costly. On the other hand, there have been many attempts to make machines learn grounding relationships from their own experiences in a bottom-up manner. Broadly speaking, they include learning of image recognition and captioning [38, 39], learning of vision and language navigation [17], concept learning from multimodal experience [40], and learning of robot action generation in response to linguistic instructions [41, 11]. Although in general this approach requires much data to acquire generalizable knowledge, it has an advantage that there is no need for humans to arbitrarily design grounding form, which is costly[1].

Here, the problem is replaced by how to design the learning system. As one of the references, we can obtain hints from our own cognitive systems. In fact, humans develop various kinds of cognitive skills from their own experiences. Imitating or modeling our learning process has brought much progress in intelligent machines [43]. Specifically, in the field of cognitive robotics or cognitive developmental robotics, researchers have been implementing human-like intelligence in robots that have their own body and performing experiments in which the robots interact physically with the world and socially with other agents to develop their cognitive capabilities [6, 8, 5]. This approach has two purposes. One is to understand human intelligence in a constructive manner by actually building the models and observing their workings in experiments. The other is to practically apply the acquired knowledge to industrial products.

Obviously, humans' language development has also been investigated in this research line [4]. As one example, by using a child-size humanoid robot called iCub, Morse et al. [44] demonstrated that the strength of the association be-

[1]Specifically, in these tasks, it is not only time-consuming but also difficult from the beginning to make explicit rules that express knowledge to link symbols with referents in the real world. Some kinds of our inference process cannot be decomposed to explicit sub-rules. For example, what is the necessary and sufficient definition of "cat"? These kinds of knowledge are called "tacit knowledge [42]."

tween a word utterance by a caregiver and sensorimotor information that changes depending on the body orientation influences word learning. Moreover, in the cognitive robotics approach, some researchers describe the problem as "symbol emergence" instead of symbol grounding [5]. They think that the symbol systems to be grounded are not given a-priori; instead, they emerge from the raw "self-enclosed" experiences in the real world through social interaction. On the basis of such a philosophy, there have been many studies in which agents have no direct contact with external symbol systems and they must develop symbolic concepts or intelligence from the experience of multimodal stream [45, 5, 46].

The cognitive robotics approach for language development stands on the same thought as the usage-based theory in the field of cognitive linguistics [47]. The usage-based theory does not think that language is based on some competence that is specialized to language and unique to humans (e.g., nature), advocated by Chomsky as the universal grammar [48], but it does think that linguistic skills are developed through daily observation and use in the basis of more general cognitive competence (e.g., nurture). In analytic philosophy, Wittgenstein also said in his latter writing [49] that "*the meaning of a word is its use.*" He argued that language systems do not have explicit rules and words have no definitions that we refer to when we use them. The meaning of a word is only based on the fact that the word works appropriately to achieve some purpose in the given situation. He called this viewpoint on language the "language game." In this game, even if there are no rigid definitions of a word, its effect gives its meaning only if it works in an appropriate manner in a given situation. From this viewpoint, the meanings of words are highly context-dependent and dynamically changing. The strong relationships between linguistic skills and other sensorimotor skills have also been reported in the field of neuroscience. Hauk et al. [50] reported that passively reading action words referring to different body parts (e.g., kick, pick, and lick) activated different brain regions that were activated by actual movements of the feet, fingers, and tongue.

The following subsections review existing studies that consider the learning of grounding relationships between language and behavior of robots or simulated agents.

2.2.1 Probabilistic Modeling

One way to model the relationships between language and other modalities is to model them as probabilistic relationships [51, 52, 53, 54, 55, 56, 57, 40, 58]. For example, Inamura et al. [51] utilized hidden Markov models (HMMs) to recognize and generate human motions. In their framework, proto-symbols, which represent a specific motion pattern, emerge in the learning process. Takano et al. [59] proposed the application of canonical correlation analysis (CCA) to extract relationships between word labels and human body motions. Their experiment showed that the trained model could recognize human body motions and conversely retrieve appropriate motions given a set of labels. Takano and Nakamura [52] proposed a stochastic model that combines a natural language model and motion language model. The model learned to interpret motion patterns as sentences and also to generate motions from sentences. Nishihara et al. [40] utilized a multimodal latent Dirichlet model (MLDA) to learn object concepts that connect multimodal information consisting of co-occurring word, auditory, visual, and tactile data. Taniguchi et al. [58] also proposed a nonparametric Bayesian model that learned location concepts, which probabilistically tied human speech with self-location of a mobile robot. Through the acquired location concepts, the performance of self-location estimation and speech recognition were improved. Tellex et al. [54] also proposed a framework called generalized grounding graphs, which dynamically instantiate a graphical model depending on semantic structure of linguistic commands, and then they inferred appropriate plans for navigation and manipulation in the graph.

One advantage of probabilistic models is high intelligibility. In the case of graphical models, each node in the graph is designed as a meaningful element. Therefore, it is easy to understand what kind of inference is performed by the model. Simultaneously, the probabilistic models have the capability to explicitly model the ambiguity between language expressions and their referents as the shapes of probabilistic density functions. Whiling maintaining an explicit representation of the ambiguity or uncertainty, the model can gradually reduce it during interaction with the environment and other agents. However, some probabilistic models have to assume constraints (e.g., Gaussian distribution for the

prior) to make the problem tractable. Another disadvantage is that a model that has the capability to deal with long-time dependencies sufficiently has not been developed yet.

2.2.2 Deterministic Modeling

In the previous subsection, we reviewed the probabilistic modeling of grounding relationships. Alternatively, methods that model them deterministically also exist. One popular method is NNs [9, 60, 41, 61, 62, 10, 63, 11]. Sugita and Tani [9] proposed a trainable architecture that consists of two RNNs — one of which is for language and the other is for robot behavior — with a small number of shared nodes called parametric bias (PB). The model learned to embed the relationships between language and behavior as topological organization in the PB space. Ogata et al. [41] employed a similar architecture to learn the bidirectional mapping between language and robot behavior. Heinrich et al. [64] proposed a model that connected three RNNs. Each RNN was specialized for vision, proprioception, and language, respectively, but they connected with each other. After learning, the model could generate sentences that described robot motions as a sequence of characters. Stramandinoli et al. [11] utilized a Jordan-type RNN [24] to ground abstract words (e.g., use and make) in robots' sensorimotor experiences. The abstract words were learned by recalling the meanings of previously learned basic words and combining them. Hinaut et al. [63] used an echo state network [65] to learn bidirectional mapping between sentences and their predicate–argument representations through human–robot interactions.

NNs are deterministic system, but there are some ways for them to deal with the ambiguity of language. One way is to shape the output in a certain layer into a form of probability distribution. For example, in the case of image classification, by activating the output layer by the softmax function, the model predicts the posterior distribution on the classes. Also, in the case of image captioning, although RNN models generate captions deterministically by the greedy method, they can also generate different sentences in a probabilistic manner by sampling from the predicted probability distribution. Moreover, there is a way to replicate probabilistic processes by perfectly deterministic NNs. For example, Tani and

Fukumura [66] demonstrated that a deterministic RNN can regenerate a symbol sequence that follows a simple probabilistic rule by self-organizing a chaotic dynamical system. Namikawa et al. [67] also showed that an RNN can learn to replicate pseudo-stochastic transitions between multiple robot motion primitives by a similar chaotic structure.

An advantage of NNs is that by introducing recurrent connections and some gating mechanism such as long short-term memory (LSTM) [68, 69], it can achieve much higher performance in learning temporal structure with long-term dependency without a-priori knowledge. One disadvantage of NNs is that it is difficult to understand their behavior since their representations in hidden layers are in a distributed form. Therefore, they are sometimes called black-boxes. Recently, there have been many studies that propose methods to visualize the internal behavior of NNs [70, 71] and to make their representations more intelligible [72, 73].

2.3 Recent Trend: Methods of Deep Learning

A recent trend in grounding learning is methods that employ deep learning. Started by the brilliant success in the ILSVRC2012 [38], which is an international competition for visual recognition, deep learning has been applied to various fields including speech recognition [74, 75] and synthesis [76, 77], NLP applications [78], and robotics [79, 80] and they outperform conventional methods. Thanks to their deeply stacked non-linear transformation, deep learning models have a much stronger capability to (1) extract important features from raw data, (2) recognize patterns using the extracted features, and (3) generate new samples from their latent representations. Their training has been successfully performed owing to the large size of datasets, sophisticated optimization techniques for gradient descent [81, 82], and the great computational power provided by GPU machines. Most recently, deep learning has also been used for learning grounding relationships between language and behavior of intelligent agents [15, 16, 17, 83, 84]. Hermann et al. [15] and Chaplot et al. [16] trained their deep neural network models by reinforcement learning so that an agent in a simulated environment navigates to objects indicated by linguistic instructions. Anderson et al. [17] also

trained a deep neural network model that combined a convolutional neural network (CNN) and an LSTM-RNN to learn a navigation task in an environment that traced real-world indoor scenes. Hughes et al. [85] also applied deep reinforcement learning to a task that rearranged objects in a cell-grid world in response to linguistic instructions. Tung et al. [83] learned the relationships between linguistic instructions and goal states for a robot to execute tabletop tasks. Their model learned action policy by utilizing the acquired relationships between instructions and goal states as rewards. Suzuki et al. [84] used a CNN-RNN model for a shirt-folding task conducted by a robot. The model learned the relationships between word signals and the order of folding operations in a supervised manner.

Many of these systems adopted EDM or similar kinds of architecture, but they have mainly dealt with a unidirectional translation, i.e., only from language or to language.

2.4 Position of this Thesis in Terms of Related Works

This chapter has reviewed existing studies from the following points of view.

- How the earliest artificial intelligence (i.e., rule-based) modeled the cognitive tasks computationally and how it failed.
- What kinds of learning-based model have been proposed to ground language in robot behavior in the field of cognitive developmental robotics.
- How recent deep learning methods are applied to grounding learning.

The contribution of this study is three-fold. The first one is to apply an EDM architecture to end-to-end learning of the mapping between language and behavioral sequence. The EDM was firstly introduced in the field of NLP for the machine translation, question answering, and chat system etc., and scored outstanding performance on these tasks thanks to its capability to learn long-term dependency. We propose to apply the EDM to conversion between language to robot behavior by extending it to integrate multimodal information together

to solve the context dependency of the meanings of sentences. By this framework, the semantic relationships between language and robot behavior conditioned on the current context are learned from data without a-priori knowledge or arbitrary rules.

Secondly, the proposed EDM is able to deal with not only grounded content words but also function words. In the field of cognitive robotics, existing studies that attempted to ground language in robot behavior by neural networks [9, 10, 11] mainly focused on content words. Although probabilistic models are also able to learn the stochastic co-occurrence relationships between content words and multimodal features, it is difficult for them to learn to model how function words combine the grounded meanings of other content words. This study deals with logic words, such as "not", "and", and "or", as examples of function words and shows that the proposed framework appropriately maps sentences that include both content words and logic words into corresponding robot behavioral sequences. Our analysis also shows that the function words are represented by an RNN in accordance with their functions as logical operators.

The third contribution is the building of a computational model that bidirectionally converts between language and behavior by extending the EDM to a two-coupled form. Most of the conventional studies on learning between language and robot behavior have mainly dealt with only unidirectional conversion. Recently, deep learning methods have been used for language grounding tasks in simulated environments [15, 16] and have achieved outstanding performances. However, in general, they are also optimized for a unidirectional task from input to output, i.e., from language instructions to action sequences. There have also been studies on cross-modal retrieval, where the paired representations of different modalities (e.g., video, sound, and text) in deep NNs were bound with each other to organize semantically similar representation spaces among modalities [20, 21, 22]. Through such shared representations, data samples in each modality can be retrieved from ones in the other modalities. However, they dealt with retrieval only, so they did not generate novel tokens from the representations. On the other hand, most recently, there have been studies on bidirectional mapping between different modalities using deep variational autoencoders (VAEs) [86, 87]. A VAE

is a neural generative model that can generate data samples conditioned on latent representations ($\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z})$) and also infer the posterior distribution of the latent representation $q(\boldsymbol{z}|\boldsymbol{x})$ given a data sample by variational approximation. By extending a VAE in a multimodal architecture, mutual inference between modalities can be achieved. However, these studies focused only on mapping between static modalities (e.g., images, binary attributes, and categories), not on temporal data such as video or motions. In the current study, by combining two RNN-based EDMs and binding their representations, we achieve bidirectional conversion between temporal data, i.e., from linguistic expressions to robot behavioral sequences and vice versa, through the shared representation.

Chapter 3

RNN-based Encoder–decoder Model to Ground Language in Robot Behavior

3.1 Introduction

In this chapter, we propose a method that unidirectionally converts linguistic expressions (resp., robot behavioral sequences) into robot behavioral sequences (resp., linguistic expressions). As mentioned in Chapter 1, this study addresses the following three requirements:

- **Requirement 1 (R1):** Solving the context dependency of meanings of linguistic expressions.
- **Requirement 2 (R2):** Understanding both content words and function words.
- **Requirement 3 (R3):** Bidirectional conversion between language and behavior.

The system proposed in this chapter addresses R1 and R2. We will address R3 in Chapter 6. The method described here combines the earlier mentioned A1 and A2. We propose the utilization of **an RNN-based EDM (A1)** that converts linguistic instructions into robot behavioral sequences (and vice versa) through

a fixed-dimensional latent representation space. The model learns to solve **the context dependency of language(R1)** by receiving a **multimodal input stream (A2)**. Moreover, RNNs have the capability to transform their internal state non-linearly depending on an external input and to self-organize internal dynamics that reflect highly structured temporal patterns of data. Thanks to these capabilities, **function words are also encoded by the model together with content words (R2)** in accordance with their specific functions. For example, "don't," which directs a robot to generate behavior in an opposite manner, works as a non-linear transformation to embed orthogonal sentences into the same area in the latent representation space. "Or," which requires behavior production that looks apparently random, is represented as unstable areas of the RNN's dynamical system.

The next section describes the general form of time series data processing by RNNs. We also review the original EDM used in the field of natural language processing for machine translation, question answering, etc. After the review, we present our method in Section 3.3 and 3.4.

3.2 Time Series Data Processing by Recurrent Neural Networks

This section describes the working of a general type of RNN. An RNN (Figure 3.1 [B]) is a model that adds recurrent connections to the hidden layer of a feed forward NN (FNN; Figure. 3.1 [A]). FNNs transform an input vector into an output vector through one or multiple hidden layers, each of which consisting of a trainable affine transformation and non-linear activation function (e.g., tanh, softmax, or ReLU). In general, FNNs simply map an input data point into another data point[1]. Therefore, they cannot deal with data that have sequential or temporal patterns[2]. In contrast, context layers of RNNs receive their own previous output through recurrent connections (orange arrow in Figure 3.1 [B]). The input history

[1]The dimension of an output vector can be different to that of an input vector.

[2]There are several ways to deal with temporal patterns in the FNN architecture. For example, time delay neural networks [88] have a FNN structure and are able to capture temporal patterns of data by receiving a concatenation of multiple vectors in a fixed-length time window at once.

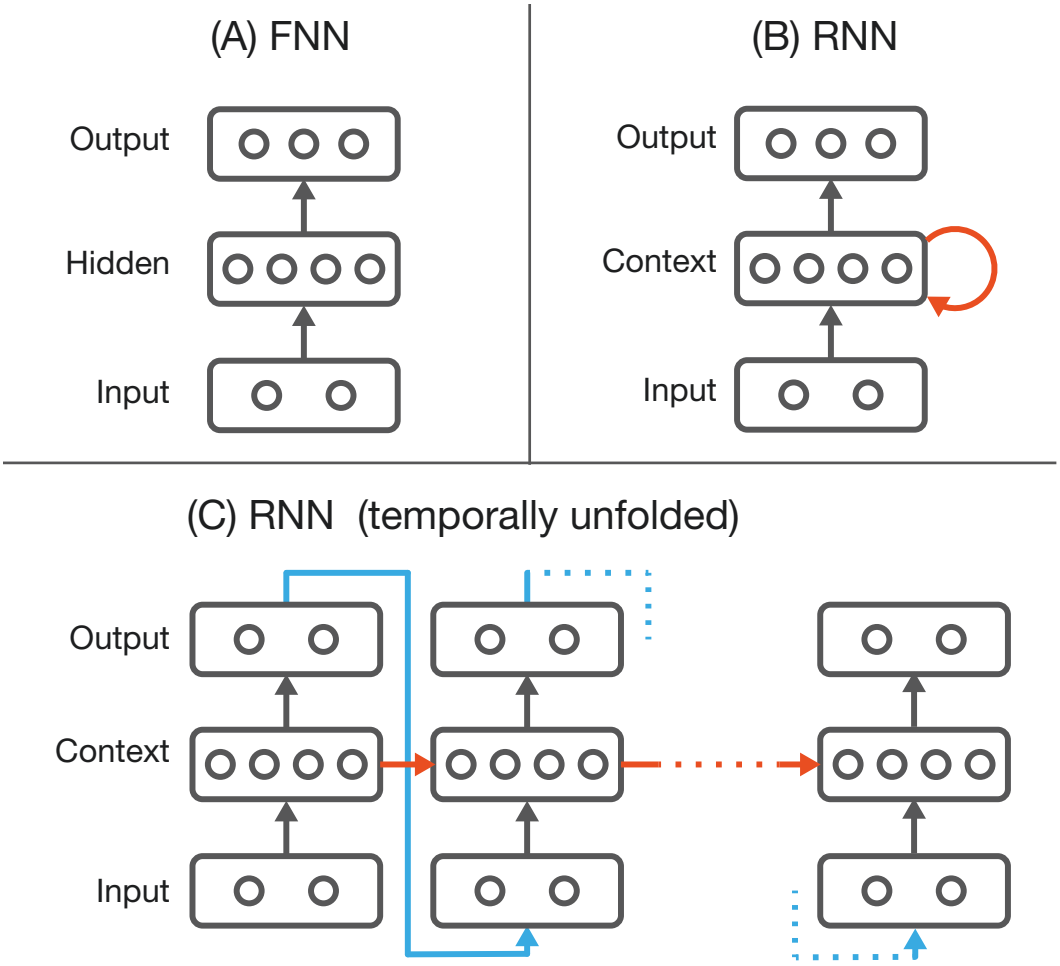

Figure 3.1. (A) FNN with one hidden layer. (B) RNN with one context layer. The orange line represents the recurrent connection. (C) RNN. Time is represented in the horizontal direction by unfolding the recurrent connection. There are cases that output vectors are recursively fed into the input layer at the subsequent time step, as shown by the blue lines.

is encoded together with the current input into the fixed-dimensional space of the context layer. Thanks to such an architecture, they can learn the temporal structure of data. Figure 3.1 [C] visualizes the same RNN by unfolding the recurrent loop horizontally.

Now, let sequential data $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T)$ be fed into an RNN one by one. At each time step, the output of the context layer of the RNN $\boldsymbol{h}_t$ is calculated as follows:

$$\boldsymbol{h}_t = \mathrm{RNNCell}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}). \tag{3.1}$$

In general, RNNCell is a non-linear function. In the case of a normal RNNCell, the update is calculated as follows:

$$\boldsymbol{c}_t = \boldsymbol{W}_{\text{in}}\boldsymbol{x}_t + \boldsymbol{W}_{\text{c}}\boldsymbol{h}_{t-1} + \boldsymbol{b}, \tag{3.2}$$

$$\boldsymbol{h}_t = f(\boldsymbol{c}_t). \tag{3.3}$$

Here, $\boldsymbol{W}_{\text{in}}$ and $\boldsymbol{W}_{\text{c}}$ are trainable matrices, $\boldsymbol{b}$ is a trainable bias term, and f is an activation function. $\boldsymbol{c}_t$ is called an internal state, a cell state, or a context state. In addition, some sophisticated cells, such as gated recurrent units (GRU) [89] and LSTMs [69], have a gating mechanism. Therefore, they can explicitly filter out unnecessary input, maintain or forget contextual information, and control whether the current cell state is fed into the output. The output is recursively fed into the layer itself at the next time step. If a following layer exists, the output is also fed into it at the current time step.

By sequentially embedding a series of data $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T)$, the RNN attains a fixed-dimensional representation of the data as $\boldsymbol{h}_T$ (or $\boldsymbol{c}_T$). In other words, the RNN can work as an encoder of sequential data. On the other hand, the RNN can also work as a decoder that generates sequential data from fixed-dimensional representations. Given a meaningful initial context state $\boldsymbol{h}_0$, by incrementally decoding it, the RNN generates a series of vectors on the output layer. The output vectors can also be fed into the input layer in the next time step, as shown by the blue lines in in Figure 3.1 [C]. Therefore, the RNN can produce a sequence by itself without any external input.

By combining these functions, RNNs can be applied to translation between different languages [25, 26]. As shown in Figure 3.2, the encoder RNN embeds an English sentence as a fixed-dimensional representation vector. Then, the decoder RNN generates a corresponding French sentence by decoding the representation. The learning of all the learnable parameters in the encoder and the decoder is achieved based on the gradient descent method. The loss function L is defined on the output layer of the decoder as the distance between the output and the ground truth. In general, RNNs are designed so that derivatives of L at all the learnable parameters can be calculated. This framework for translation between sequences,

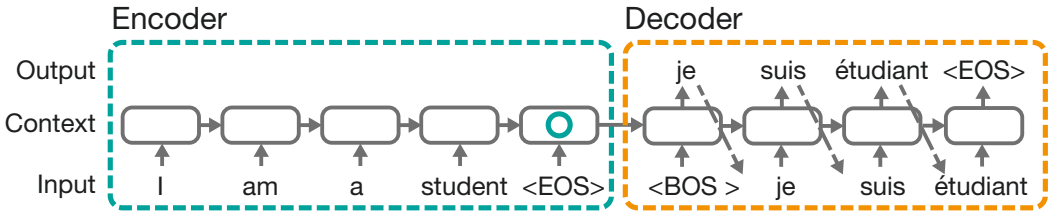

Figure 3.2. Architecture of a normal EDM. During the encoding phase, the encoder RNN incrementally encodes a source sequence by receiving an element (e.g., word) at each time step. The source sequence is projected onto the fixed-dimensional space of the context layer (indicated as a green circle). After that, the decoder RNN receives this embedding and then generates a corresponding sequence in the target domain by incrementally decoding the representation.

which consists of an RNN encoder and an RNN decoder, is called the EDM. This study applies it to the conversion learning between language and robot behavior.

3.3 Converting Linguistic Instructions into Robot Behaviors

This section describes our proposed method to convert language into robot behavior. We propose two different EDMs. The first one is directly based on the normal EDM [25], which has two separated RNNs: an encoder and a decoder. The other one does not have clear separation of an encoder and decoder. Instead, it has a single RNN that learns to internally functionalize encoding and decoding. In this case, encoding the current input and decoding the internal representation are constantly performed at each time step. Whenever an instruction is received, the information is encoded together with the current multimodal input. After the instruction reception, its representation is immediately decoded as a trajectory of joint angles.

3.3.1 Model 1: Structural EDM

First, we introduce a model that has clear separation of an encoder and a decoder. We call this model structural EDM (S-EDM). Fig. 3.3 shows a schematic diagram of S-EDM. The S-EDM consists of an encoder RNN and a decoder RNN. The encoder RNN embeds a linguistic instruction, namely a word sequence

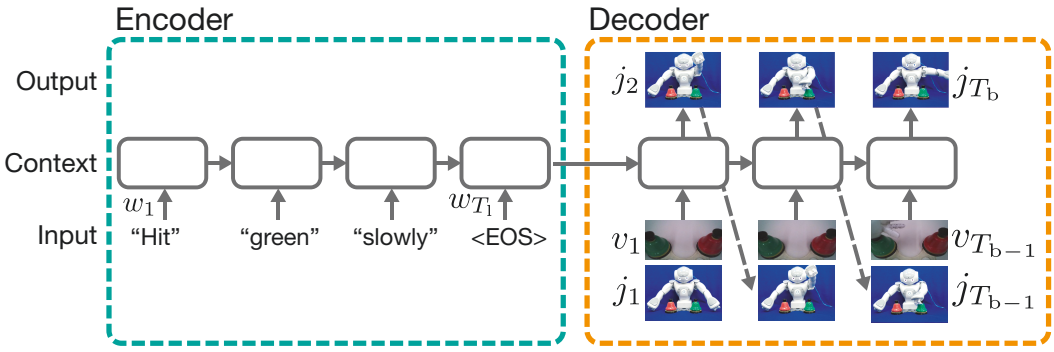

Figure 3.3. The overview of S-EDM to convert linguistic instructions into robot behavioral sequences. First, the encoder RNN embeds an instruction into a fixed-dimensional vector. The cell state after receiving <EOS> is copied to set as the initial recurrent input of the decoder. From this state, the decoder produces a behavioral sequence corresponding to the instruction. Here, the appropriate trajectory depends on the current context. The decoder solves this issue by receiving visual and proprioceptive input.

$(\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_{T_l})$, into a fixed-dimensional feature vector. At each time step, the output of the context layer is calculated as follows:

$$\boldsymbol{h}_t^{\text{enc}} = \text{EncRNNCell}(\boldsymbol{w}_t, \boldsymbol{h}_{t-1}^{\text{enc}}) \quad (1 \leq t \leq T_l). \tag{3.4}$$

$\boldsymbol{w}_t$ and $\boldsymbol{h}_t$ are a word input and the context layer output at time step t, respectively. EncRNNCell is a learnable recurrent cell, although the detail of the update calculation depends on the type of the cell (i.e., normal, GRU, LSTM, etc.) Given an input word sequence, its embedding by the encoder RNN is attained as $\boldsymbol{h}_{T_l}^{\text{enc}}$ by incrementally calculating Eq. 3.4. Next, the initial recurrent input of the decoder $\boldsymbol{h}_0^{\text{dec}}$ is set by copying the final output of the encoder $\boldsymbol{h}_{T_l}^{\text{enc}}$:

$$\boldsymbol{h}_0^{\text{dec}} = \boldsymbol{h}_{T_l}^{\text{enc}}. \tag{3.5}$$

Finally, the decoder RNN generates a behavioral sequence by recursively decoding the representation as follows:

$$\boldsymbol{h}_t^{\text{dec}} = \text{DecRNNCell}(\boldsymbol{v}_t, \boldsymbol{j}_t, \boldsymbol{h}_{t-1}^{\text{dec}}) \qquad (1 \leq t \leq T_b - 1), \tag{3.6}$$

$$\boldsymbol{j}_{t+1} = g(\boldsymbol{W}\boldsymbol{h}_t^{\text{dec}} + \boldsymbol{b}) \qquad (1 \leq t \leq T_b - 1), \tag{3.7}$$

where $\boldsymbol{v}_t$ and $\boldsymbol{j}_t$ are visual input and joint angle input at time step t, respectively.

DecRNNCell is a learnable recurrent cell. $\boldsymbol{j}_{t+1}$ is obtained by applying an affine transformation to $\boldsymbol{h}_t^{\mathrm{dec}}$ and then activating it by g. The parameters of the affine transformation — $\boldsymbol{W}$ and $\boldsymbol{b}$ — are also learnable. The decoder's own output $\boldsymbol{j}_{t+1}$ is recursively fed into the input layer at each step.

In this way, by receiving the current visual and proprioceptive information together, the decoder generates a behavioral sequence depending not only on the instruction embedding but also the current context. Learning is performed as minimization of the error between an output sequence and a target sequence $(\hat{\boldsymbol{j}}_1, \hat{\boldsymbol{j}}_2, ..., \hat{\boldsymbol{j}}_{T_\mathrm{b}})$. In other words, the following loss function L is minimized:

$$L = \frac{1}{T_\mathrm{b} - 1} \sum_{t=1}^{T_\mathrm{b}-1} ||\boldsymbol{j}_{t+1} - \hat{\boldsymbol{j}}_{t+1}||_2^2. \tag{3.8}$$

Although here we defined the loss as mean squared error (MSE) between the output and the target, we can choose other types of loss in accordance with tasks[3]. As mentioned above, derivatives of L at all the learnable parameters can be calculated by applying the back propagation through time (BPTT) algorithm [90]. Thus, the parameters can be iteratively optimized by the gradient descent method.

3.3.2 Model 2: Functional EDM

Secondly, we propose a model that does not have clear separation between the encoder and the decoder. We call this model functional EDM (F-EDM; Figure 3.4). In the case of the S-EDM, we assume that we have an explicit paired dataset of language and behavior (accompanied by a vision stream). However, we presuppose that we have only sequential data that are recorded as directly representing temporal flows of interaction, where instructions and behavioral responses are included (Figure. 4.3 is an example). To learn this form of data, the F-EDM does not have a separate encoder and decoder nor does it explicitly separate the encoding and decoding phases; it has only one RNN module. At each time step, this RNN constantly receives words $\boldsymbol{w}_t$, vision $\boldsymbol{v}_t$, and joint angle inputs $\boldsymbol{j}_t$ and generates the joint angles $\boldsymbol{j}_{t+1}$ at next time step.

[3]For example, in the case of generating word sequences, the cross-entropy loss would be appropriate.

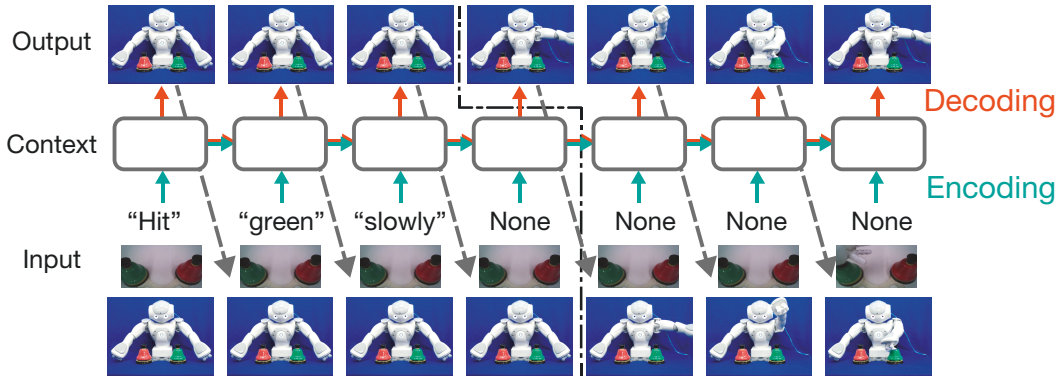

Figure 3.4. Overview of F-EDM. The F-EDM does not have a separate encoder and decoder; instead, it has only one RNN. Encoding and decoding themselves are learned as functions from the experience of predictive learning.

In this predictive learning, the F-EDM learns not only the grounding relationships between language and behavior but also the progress pattern of interaction. For example, in the case of the data shown in Figure 4.3, the imposed task consists of a stand-by phase, instruction reception phase, and behavior generation phase. Therefore, the model must learn to maintain the initial pose during the stand-by phase and instruction reception phase by generating the joint angle values corresponding to the initial pose. After receiving the instruction, the model must generate appropriate behavior immediately. Here, even while generating a behavioral sequence, the input nodes for words receive a zero vector representing that no word is currently given. After the behavior generation, the model must wait for a subsequent instruction again. In this way, the model must learn not only the relationships between instructions and behavioral responses but also the task progress pattern. After training, if only the model calculates its forward propagation, it can continue the interaction as if it flexibly switched between the instruction reception, behavior generation, and stand-by phases in appropriate contexts.

3.3.3 Comparison of S-EDM and F-EDM

Figure 3.5 compares the workings of S-EDM and F-EDM by focusing on their internal representations. In the case of S-EDM, the encoder first embeds an instruction as the final output of the context layer. After that, the decoder generates

a corresponding behavioral sequence by decoding the embedding, conditioned on visual and proprioceptive information. The switch between the encoding and decoding phases is externally given. Because of this architecture, S-EDM can be used when a parallel dataset of behavioral sequences and their corresponding linguistic sequences exists and the separation of the encoding and decoding phases can be given externally. In the S-EDM, the encoder and decoder are clearly separated, so each module is optimized in accordance with its specific function.

In contrast, in the case of the F-EDM, the interaction flow that consists of instruction reception and behavior generation is directly represented as the transition of the context state of the single RNN. First, while an instruction is input to the model, the context state moves in a fixed-dimensional space in accordance with the meaning conditioned on the visual and proprioceptive context (represented as colored solid lines in Figure 3.5 [B]). After receiving the instruction, the RNN generates a behavioral sequence while its context state further moves from the activation corresponding to the instruction embedding (represented as colored broken lines in Figure 3.5 [B]). Finally, the context state converges back into the state that represents a waiting posture, and then waits it for a subsequent instruction. In this way, the F-EDM learns to represent the repetition of instructions and behavior responses as cyclic transitions of the context state. The explicit separation between the encoding and decoding phases does not exist; instead, the task progress pattern that is implicitly included in the target data is also learned as the RNN's internal dynamics.

In practice, task progress patterns that are different from simple repetition of instruction, behavior, and waiting also occur in real interaction with humans. For example, the robot state during waiting could be different every time. There might also be cases in which the robot must refer to the previous episodes (e.g., demonstrative pronouns). Moreover, it should be possible that a human utters an additional instruction during the robot's behavior generation. Manually modeling all of these interaction progress patterns is as difficult as the grounding problem itself. Although we do not deal with all of these examples in the current study, F-EDMs have the potential to learn such varying kinds of task progress patterns that are implicitly included in training data.

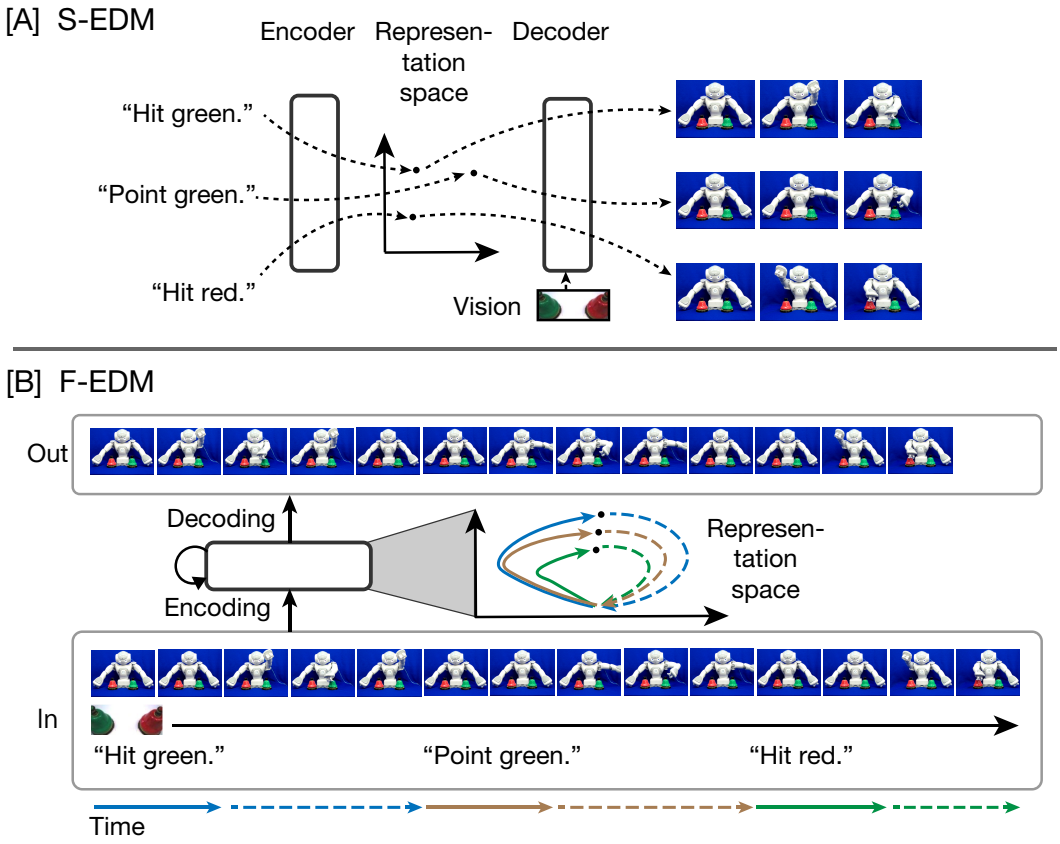

Figure 3.5. [A] In the case of the S-EDM, the encoder first projects an instruction onto the representation space. After that, the decoder generates a corresponding behavioral sequence by decoding the representation, conditioned on visual and proprioceptive information. The switch between the encoding and decoding phases is externally given. [B] In the case of the F-EDM, the interaction flow that consists of instruction reception and behavior generation is directly represented as the transition of the context state. First, while an instruction is input to the model, the context state moves in accordance with the meaning (colored solid lines). After receiving the instruction, the RNN generates a behavioral sequence while its context state further moves from the activation corresponding to the instruction embedding (colored broken lines). Finally, the context state converges back again into the state that represents a waiting posture, and then it waits for a subsequent instruction. An explicit separation between the encoding and decoding phases does not exist; instead the F-EDM learns to represent the task progress pattern, namely repetition of instruction receptions and behavior responses, as cyclic transitions of the context state.

3.3.4 How the System Solves the Problems

The method proposed here integrates A1 and A2, i.e., an EDM from language to robot behavior (A1) that also learns to merge multimodal information (A2). This method learns the grounding relationships between language and behavior

without a-priori knowledge from target data that include (explicit or implicit) pairs of them by embedding the grounding relationships as fixed-dimensional space of the context layer. Although the relationships between language and behavior depend on the current situation, by receiving visual and proprioceptive input and integrating it together with word input, an appropriate behavioral sequence can be generated (R1). Moreover, this framework can deal with both content words and function words. In the field of formal semantics, a meaning of a sentence is given as a function of its element words. How to combine meanings of words to build a meaning of a sentence is theorized by human in a top-down manner. In contrast, in the case of RNN learning, the function to build a representation of a sentence is learned from data in a bottom-up manner. For example, content words are semantically encoded together with multimodal information depending on the current situation. On the other hand, function words can contribute to build the meaning of a sentence by further transforming the representations of the content words. For example, as shown later in Chapter 5, "don't," which directs a robot to generate the inverse behavior, works as a non-linear transformation of the context state.

3.4 Converting Robot Behaviors into Descriptive Sentences

The previous section described the EDM, which unidirectionally converts language into robot behavior. In the opposite direction, we can build another EDM to convert robot behavioral sequences into descriptive sentences in a similar manner, as shown in Figure 3.6. Here, we describe only the S-EDM. The encoder first encodes a behavioral sequence that consists of a series of joint angles and accompanying visual information. Then, the final output of the encoder is copied to set as the initial recurrent input of the decoder. Finally, the decoder generates a descriptive sentence by decoding the representation.

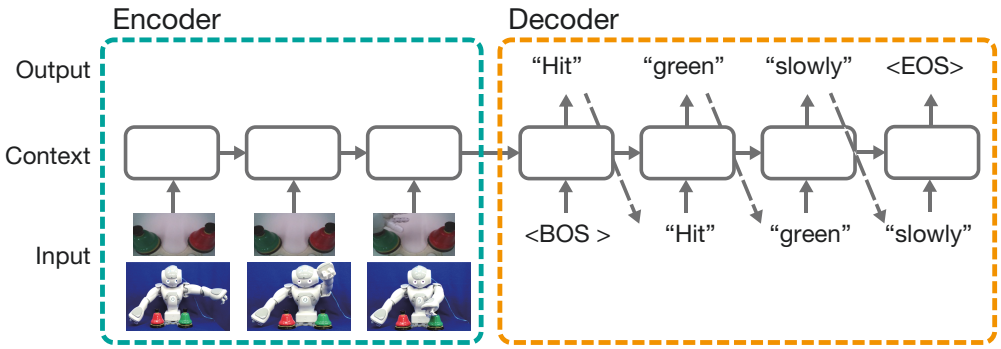

Figure 3.6. Overview of the S-EDM, which converts robot behavioral sequences into linguistic descriptions.

3.5 Summary

In this chapter, we proposed RNN-based EDMs that learn to convert linguistic expressions (resp., robot behavioral sequences) into robot behavioral sequences (resp., linguistic expressions) depending on the current context given as multimodal input. We introduced two model variations — S-EDM and F-EDM. The S-EDM is trained by explicit paired data of language and behavior. The grounding is achieved through fixed-dimensional latent representation. On the other hand, the F-EDM is trained with sequential data that directly represent an interaction flow and implicitly include pairs of language and behavior. From the experience of predictive learning, the F-EDM learns to represent the interaction flow as its internal dynamics and the grounding is also achieved in the middle of the dynamics. The topics discussed in this chapter are summarized in Figure 3.7.

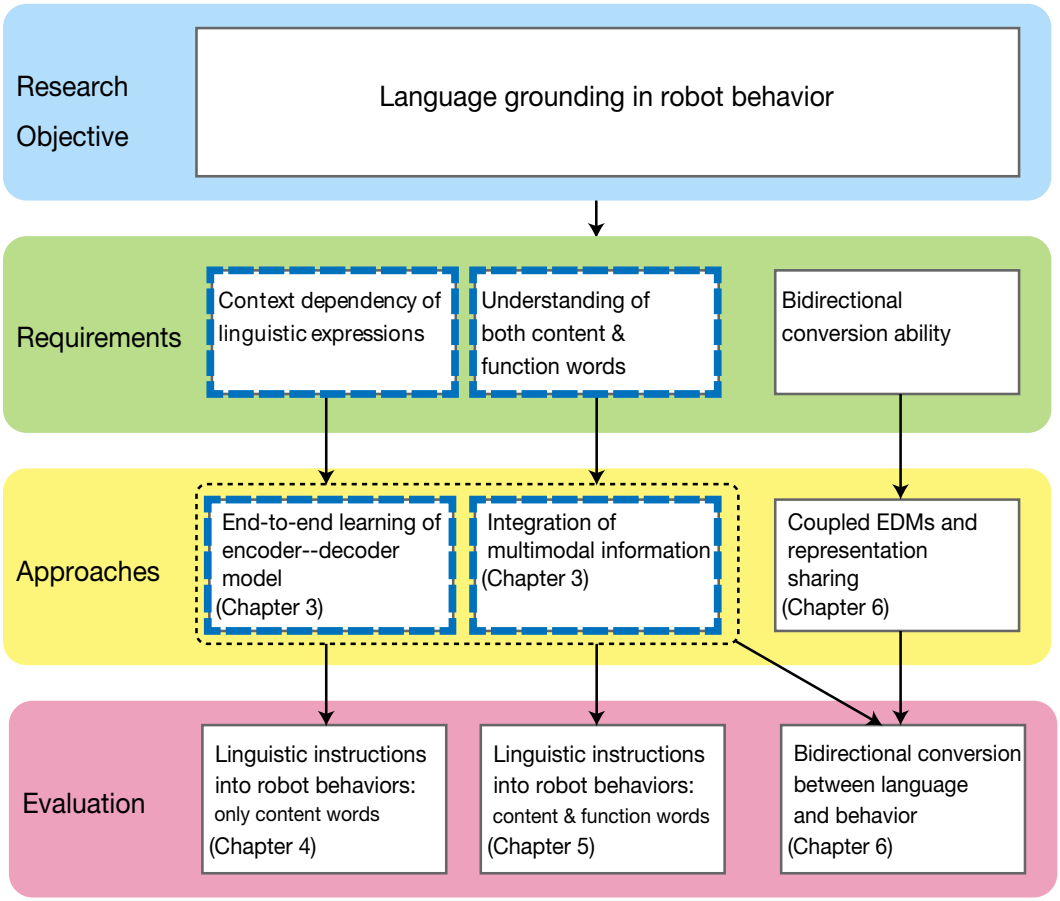

Figure 3.7. The main targets discussed in Chapter 3.

Chapter 4

Grounding Context Dependent Instructions in Robot Behaviors

4.1 Introduction

Chapter 3 proposed EDMs that are based on the RNN architecture to unidirectionally convert language to robot behavior or behavior to language. In this chapter, to evaluate the conversion capability of the proposed system, we perform an experiment using a real robot. The main target that we focus on here is **the context dependency of the relationships between language and behavior (R1)**. The experiment performed here evaluates the conversion performance of the F-EDM from multi-word instructions to robot behavioral sequences. The instructions consist of three or four content words including at least one verb, object, and adverb. The meanings of instructions are determined by the combination of these element words and also depend on the current context that is given as visual input.

In conventional methods in general, to convert language instructions into robot behavior, one must manually separate the problem into several sub-problems, such as parsing a sentence, identifying the target object, planning a motion, etc, which is a costly task. In contrast, our proposed framework learns the context-dependent relations between linguistic instructions and behavioral responses from data in an end-to-end manner. This chapter demonstrates that the F-EDM can learn to convert linguistic instructions into behavioral sequences by compositionally

encoding element words in a way that is integrated with the visual information and by appropriately decoding the representation. Note that we do not deal with function words but only content words in this chapter.

The remainder of this chapter is organized as follows. In Section 4.2, we explain the detailed architecture of the model employed. Section 4.3 describes the experimental design, such as the details of the task imposed on the robot and the target data for training. After that, we report the learning results. In Section 4.4, we report the achieved task performance. In Section 4.5, we analyze how the RNN learns to internally represent grounding relationships. In Section 4.6, we state the pros and cons of the proposed method in terms of our experimental results and present directions for future work. Section 4.7 concludes this chapter.

4.2 Learning Model Details: F-EDM Based on Multiple Timescale RNN

Here, we describe the details of learning model employed. As shown in Figure 4.3, the dataset employed here is not explicitly separated pairs of instructions and behavioral sequences, but it directly represents episodic flows of an instruction followed by a behavioral response. To learn this form of target data, we use the F-EDM. As mentioned in Chapter 3, the F-EDM has no explicit separation of an encoder and decoder and is instead built as a single RNN architecture. The model constantly receives the current word, vision, and joint angle input and then predicts the joint angles for the future time step.

In this experiment, we employ an NN called a multiple timescale RNN (MTRNN). It stacks a couple of context layers that work with different time constants (Figure 4.1). Thanks to this constraint, MTRNNs acquire functions that work in different time scales on different layers in a hierarchical manner ([91, 92]). For example, Yamashita and Tani [91] demonstrated that an MTRNN stacking two context layers successfully learned robot arm motion sequences that were comprised of multiple reusable motor primitives. One context layer worked with a small time constant and the other worked with a large time constant. Through learning, the model organized its dynamical system that coupled with the mo-

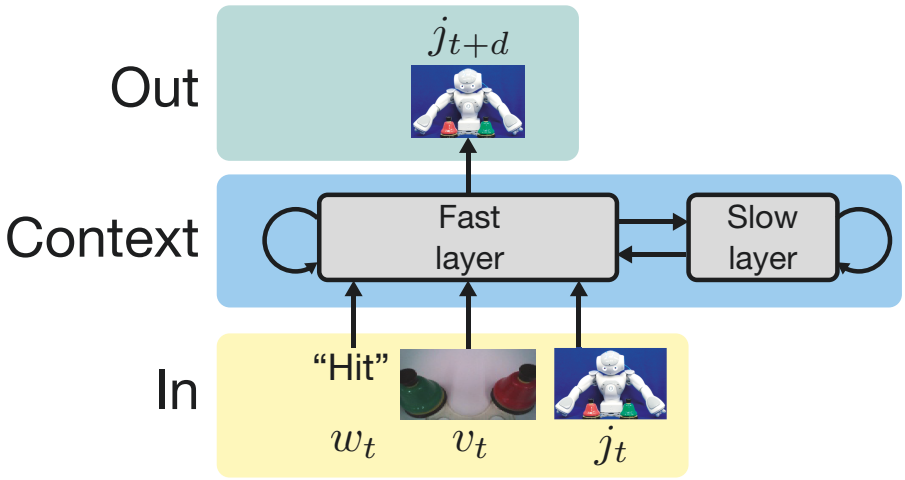

Figure 4.1. Two-layer MTRNN. The I/O and slow layers do not have direct connection, but input signals are fed into the slow layer through the fast layer, and the slow layer activation also controls the output through the oppositely directed connection.

tions to be generated in a temporally hierarchical manner. The layer with the small time constant organized dynamics that worked synchronously with the motor primitives, and the layer with the large time constant represented the orders of primitives.

In the current task, a two-layered MTRNN enables the robot to cope with both the global context of interaction and the current complicated I/O flow. The grounding relationships between sentences and behaviors and the global context of the interaction are represented in the slow layer, which works with a large time constant. On the other hand, the fast layer, which works with a small time constant, organizes representations that more directly correspond to the details of the current I/O flows. To be more precise, the fast layer facilitates the two-way transformation between the I/O flow and the abstract representation of global context of the task progress in the slow layer. While a language command is given, the fast layer simultaneously receives visual features, integrates them with word information, and then feeds the encoding into the slow layer. The slow layer receives the integrated information and obtains appropriate activation in accordance with the meaning of the instruction in the current context. This is a bottom-up process. On the other hand, during behavior generation, the representation in the slow layer is decoded as various joint angle sequences on the output layer through the

fast layer. This is a top-down process. By being functionalized hierarchically, the model enables the robot to execute this interactive grounding task. The details of the MTRNN formulation are described in Appendix A.1.

To train the MTRNN-based F-EDM, we apply two learning techniques. One is to learn to predict not only joint angles but also other modalities together. Some studies have empirically reported that by predicting multiple modalities that are not mandatory for the task accomplishment together as an auxiliary task, the performance on the main task can be improved [15, 93]. The other technique is learning to predict not only the next target state but also its uncertainty. This technique was proposed independently in both [94] and [95]. Murata et al. [95] reported that when target data include unpredictable noise, learning to explicitly predict the target state in the form of its mean and variance can improve the learning stability. The loss function defined in this learning scheme is described in Appendix A.2.

4.3 Experimental Setup

4.3.1 Task Design

This subsection describes a grounding task that the model learns to execute. First, we put two bells whose color is red, green, or blue in front of the robot to its left and right. After that, we give the robot a three-word instruction that is comprised of a verb, a position word, and an adverb (e.g., "ring right fast"; referred to as P-type) or a verb, a color word, and an adverb (e.g., "point green slowly"; referred to as C-type). Here, the object words indicate one of the colored bells. Adverbs direct the speed of the motion. In the cases where the two bells are painted in the same color, the robot cannot recognize which bell is referred to by a C-type sentence. Thus, a four-word instruction that is comprised of a verb, a color word, a position word, and an adverb (e.g., "ring red left slowly"; referred to as C′-type) is used instead. After receiving an instruction, the robot immediately begins to produce a behavioral sequence that corresponds to the instruction. After generating a behavioral sequence, the robot waits for the next instruction. We refer to this chunk consisting of instruction reception, behavior

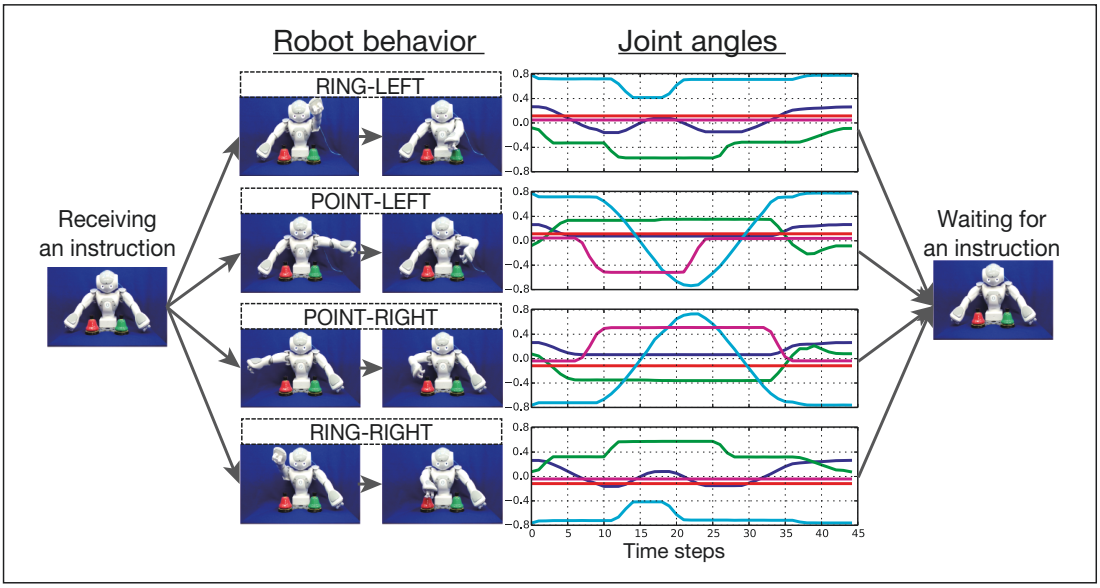

Figure 4.2. Illustration of behavioral sequences. Each of them can be produced at both FAST and SLOW speed. For each behavior, we plotted only the five joints on the moving arm.

generation, and stand-by as an "episode." This task includes nine possible bell combinations (R,G,B) × (R,G,B), eight possible behavioral patterns (POINT[1], RING) × (LEFT, RIGHT) × (SLOW, FAST), and two possible instruction types (P, C or C′), which means there is a total of 9 × 8 × 2 = 144 possible episode patterns.

4.3.2 Target Data

We next describe the target sequential data that represent this grounding task. The target data were collected through the following process. First, we collected behavioral sequences using a real robot (Figure 4.2). We recorded the behavioral sequences as a sequence of 10-dimensional vectors, whose elements correspond to different robot arm joints, by running programs that control the robot arm joints along predesigned trajectories. The sampling interval was 240 ms. FAST and SLOW behavioral sequences took approximately 30 and 45 time steps, respectively. After recording, we scaled the movable range of each joint into [-0.8, 0.8]. At the same time, we recorded images from a built-in camera on the robot

[1]In this thesis, we denote specific behaviors that the robot executes and their elements with capital letters.

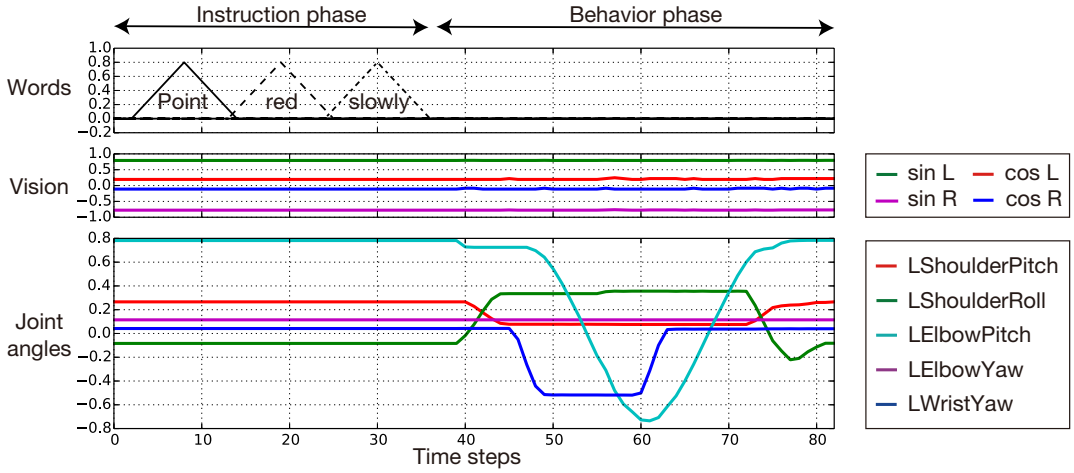

Figure 4.3. An episode example. In this episode, we give the instruction "point red slowly" to the robot in a context in which the red and green bells are located on the left and right side, respectively. In response to the instruction, the robot executes the POINT-LEFT-SLOW behavior. We plotted only the five joints of the left arm.

head. We then converted the recorded images into four-dimensional vectors that represent bell colors. More precisely, four elements correspond to the sine and cosine of the hues of bell colors.

After recording and preprocessing all the possible combinations of behaviors and bell arrangements, we prefixed instruction sentences to them on a computer. Sentences were constructed as a succession of nine-dimensional vectors. Each element is assigned to one word ("point", "ring", "left", "right", "red", "green", "blue", "slowly", and "fast"). An instruction consists of a series of words, each of which represented as a triangle activation that reaches the top — 0.8 — in six time steps and returns to 0.0 in six time steps[2].

In this procedure, we made all the 144 possible episode patterns as sequences of 23-dimensional vectors. Figure 4.3 illustrates an example of an episode.

[2]Here, the joint angles and vision during the instruction phase took the same values as those at the first step of the following behavior phase. Thus, the robot rests in the initial posture while receiving an instruction, and the bells are not relocated through an episode, although some fluctuation and noise could be added during data recording.

4.4 Learning Results

4.4.1 Evaluation Method

To train the model, we made sequences that concatenate multiple episodes in a random order[3]. Intervals between episodes were randomly set to at least 3 and at most 25 time steps[4]. The model extracts the task progress pattern implicitly included in the sequential data and attains the skill to switch phases autonomously. We prepared three training datasets and one test dataset as follows:

- **Train 1**: includes 72 sequences. Each of them is a concatenation of 20 episodes; all 144 possible episode patterns are included at least once (144/144).
- **Train 2**: includes 72 sequences. Each of them is a concatenation of 20 episodes; only half of the possible episode patterns are included in this dataset. The other patterns were used to evaluate generalization ability (Table 4.1; 72/144).
- **Train 3**: includes 72 sequences. Each of them is a concatenation of 20 episodes; two-thirds of the possible patterns were excluded (Table 4.2; 48/144).
- **Test**: includes 72 sequences. Each of them is a concatenation of 20 episodes; all possible episode patterns were included; the order of episodes is different from that in the above sets.

We carried out training and evaluated the resulting performances independently for each training dataset. In the test, only instructions and vision are given to the model externally. In contrast, the input nodes for joints receive values predicted by the corresponding output nodes d steps before[5]. In this forward

[3]By experiencing repetitions of multiple episodes, the capability to sustainably continue to interact multiple times is acquired. In fact, in a preliminary experiment, trained with a dataset to which such concatenation was not applied, the model could not achieve the capability of moving back to the stand-by state after behavior generation despite succeeding in learning the relationships between language and behavior.

[4]By doing this, a fixed-point attractor that enables the robot to wait for subsequent instructions is organized.

[5]In Chapter 3, we described the basic model with $d = 1$; in other words, the joint angle output is fed into the input nodes on the next time step. In fact, d can be a hyperparameter that should be controlled depending on resolution of recording frames.

Table 4.1. Episodes included in the Train 2 dataset. To direct the robot to execute a specific behavior (row) in a specific bell combination (column), two instructions are possible: P-type and C- or C′-type. In the Train 2 dataset, the model experiences one of them for each combination of behavior and bell arrangement. For instance, when green and red bells are put in the left and right side, respectively (G-R), the model experiences only the P-type sentence to execute POINT-LEFT-FAST behavior (6nd row, 4th column); when the bell combination is Red-Blue, the model experiences only the C-type sentence to execute RING-LEFT-SLOW (1th row, 3rd column).

Behavior \ Bell colors	R-R	R-G	R-B	G-R	G-G	G-B	B-R	B-G	B-B
RING-L-SLOW	C′	P	C	P	C′	P	C	P	C′
RING-L-FAST	P	C	P	C	P	C	P	C	P
RING-R-SLOW	P	C	P	C	P	C	P	C	P
RING-R-FAST	C′	P	C	P	C′	P	C	P	C′
POINT-L-SLOW	P	C	P	C	P	C	P	C	P
POINT-L-FAST	C′	P	C	P	C′	P	C	P	C′
POINT-R-SLOW	C′	P	C	P	C′	P	C	P	C′
POINT-R-FAST	P	C	P	C	P	C	P	C	P

propagation, the sequence generated by joint angle output nodes is understood as the model's autonomous behavior.

We evaluate the resulting performance by numerical simulation on a computer without using the real robot but with the Test dataset. The evaluation is performed by calculating the root MSE (RMSE) between the generated joint angle values and the ground truth values per joint per time step. In the Test dataset, the episodes are ordered differently from the datasets for training. If the model can learn the temporal patterns as a systematic grounding of sentences in robot behaviors rather than by rote memorization of whole sequences that are a succession of multiple episodes, it will be capable of generating appropriate behavioral sequences even in situations with differently ordered episodes.

4.4.2 Model Hyperparameters

The hyperparameters employed in the current experiment are listed in Table 4.3. At the beginning of learning, we set the learning rate α to 0.1 and updated it adaptively along the learning progress using an algorithm proposed in Namikawa and Tani [96].

With respect to each training dataset, learning was carried out 10 times from

Table 4.2. Episodes included in the Train 3 dataset (48/144). The model experienced neither P-type nor C-type instruction in the black-filled situations.

Behavior \ Bell colors	R-R	R-G	R-B	G-R	G-G	G-B	B-R	B-G	B-B
RING-L-SLOW	C′		C	P			C	P	C′
RING-L-FAST	P			C	P	C		C	P
RING-R-SLOW		C	P	C		C	P		P
RING-R-FAST		P	C		C′	P	C	P	
POINT-L-SLOW		C	P		P	C	P	C	
POINT-L-FAST	C′	P	C		C′			P	C′
POINT-R-SLOW	C′			P	C′	P	C		C′
POINT-R-FAST	P	C		C			P	C	P

Table 4.3. Hyperparameters of the employed model.

Hyperparameter	Value
Number of nodes in the fast layer N_B	80
Number of nodes in the slow layer N_T	30
Time constant of the fast layer τ_B	2
Time constant of the slow layer τ_T	12
Prediction constant d	4
Learning rate α	0.1
Momentum term for gradient descent η	0.9
Learning iterations	100,000

randomly initialized trainable parameters by different seeds. We evaluated the resulting performance every 5,000 iterations. The learning progress with respect to each of the datasets is shown in Figure 4.4. Each line is an error curve corresponding to one random seed. Although learning tends to develop in an unstable manner, we can observe the tendency of error decrease with respect to both experienced and unexperienced patterns. The following reports the results of the models that achieved the best performance among all the random seeds and learning epochs (orange circles in Figure 4.4). We refer to the best model trained with the Train N dataset as "Model N."

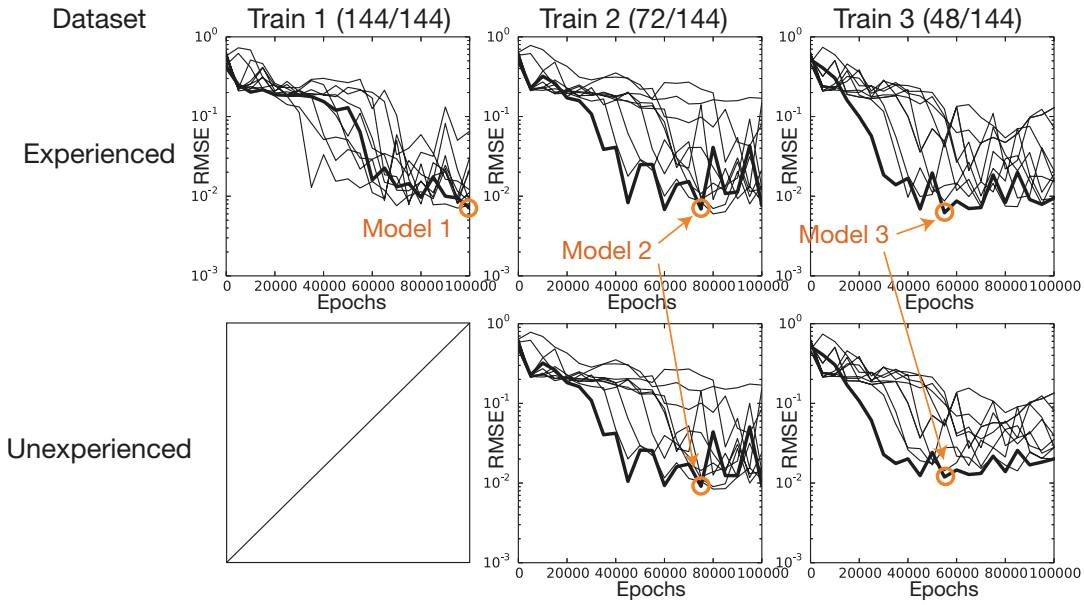

Figure 4.4. The error curves with respect to each of the datasets. Each line shows a learning development that started from differently initialized parameters. The orange circles indicate the models that achieved the best result among all the learning trials and epochs.

4.4.3 Task Performance

Behavior Generation

Table 4.4 quantitatively shows the performance results of the behavior generation. All the models succeeded in generating an appropriate behavioral sequence in all the experienced episode patterns. The overall RMSEs of Models 1, 2, and 3 during behavior generation were 7.06e-03, 6.91e-03, and 6.24e-03, respectively. Even in the worst episode, the RMSE was only 1.17e-02. Models 2 and 3 did not expe-

Table 4.4. Behavior generation performance. "Mean" indicates the overall RMSE during the test. "Worst" indicates the RMSE in the worst episode. "StdDev" indicates the standard deviation of the RMSE over all episodes.

	Experienced			Unexperienced		
	Mean	Worst	StdDev	Mean	Worst	StdDev
Model 1 (144/144)	7.06e-03	1.17e-02	1.76e-03	–	–	–
Model 2 (72/144)	6.91e-03	9.83e-03	1.50e-03	9.08e-03	2.03e-02	3.08e-03
Model 3 (48/144)	6.24e-03	7.90e-03	8.4e-04	1.19e-02	5.43e-02	7.03e-03

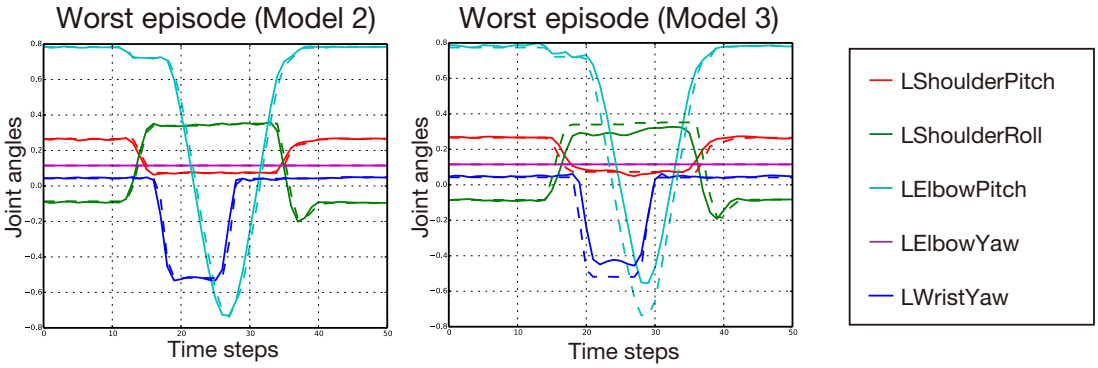

Figure 4.5. [Left] Joint angle trajectory predicted by Model 2 in the worst unexperienced episode that scored the largest RMSE (2.03e-02). The solid lines indicate predicted joint angle values, and the dashed lines are the correct ones. [Right] The worst unexperienced episode by Model 3. The RMSE was 5.43e-02. We plotted only the five joints on the left arm because in these episodes the correct behavior is the POINT-LEFT-FAST one.

rience a number of episode patterns during training. The errors in unexperienced patterns were a little larger than those in experienced patterns. However, the visual comparison confirmed that the produced behavioral sequences were rather similar to the ground truth even in the episode that scored the largest RMSE, as shown in Figure 4.5. These results demonstrate that the model achieved the generalization ability to ground instructions appropriately in behavioral responses even in the unexperienced situations by learning systematic relationships among language, vision, and behavior.

Waiting Ability

One attractive point of the F-EDM is its capability to learn the task progress pattern together with grounding. The current task contains multiple repetitions of instruction reception, behavior generation, and stand-by. If the model appropriately learned this task progress pattern, it could achieve the capability of waiting for a subsequent instruction after every behavior execution. In fact, we evaluated this capability and confirmed that after every behavior execution, both arms returned to the initial posture and stayed in that posture until the next sentence was given. The RMSE by Models 1, 2, and 3 during the stand-by phase were 5.81e-03, 3.76e-03, and 3.24e-03, respectively. We also confirmed that even when we gave an instruction after an unexperienced long time of up to 100 steps,

the robot successfully produced an appropriate behavioral sequence. In brief, the model has achieved the capability of stably waiting for instructions. In other words, the model learned not only the grounding relationships between language and behavior but also the interaction progress pattern of repetitions of instruction reception, behavior generation, and waiting for a subsequent instruction.

As demonstrated by these results, the model could continue this interactive task by reusing learned grounding relationships in appropriate contexts.

4.5 Analysis of Internal Representations

In the previous section, we exhibited that the models learned to behave appropriately in the current task. We next investigate how the RNN has learned to internally represent grounding relationships in its dynamics.

4.5.1 Slow Layer Dynamics

To investigate how the model internally represents this grounding task, we visualized the context states[6] in the both layers during the test by principal component analysis (PCA), which projects the original high-dimensional vectors into a low-dimensional subspace. In the following, we explore the representations by Model 1. In Figure 4.6, the top-left panel visualizes the time development of the context state of the slow layer during episodes in the PC1 direction. Respective lines show the representative development of one of eight different behaviors[7]. The context state of the slow layer bifurcates in accordance with the given words. Different words (e.g., point or ring) lead it to different branches. After three bifurcations (POINT–RING, LEFT–RIGHT, and SLOW–FAST), the context state reaches one of eight different embeddings that correspond to the respective behaviors. From the state that represents grounding of a sentence in a corresponding behavior, the model immediately starts to generate a behavioral sequence.

[6]Here, we visualized the vector space before applying the activation function.

[7]More precisely, respective lines are the average development over all the episodes in which one specific behavior was executed. For instance, the red solid line indicates the average over all the episodes in which POINT-LEFT-SLOW was executed. However, the episodes in which a four-word instruction (i.e., C′-type instruction) was used were excluded.

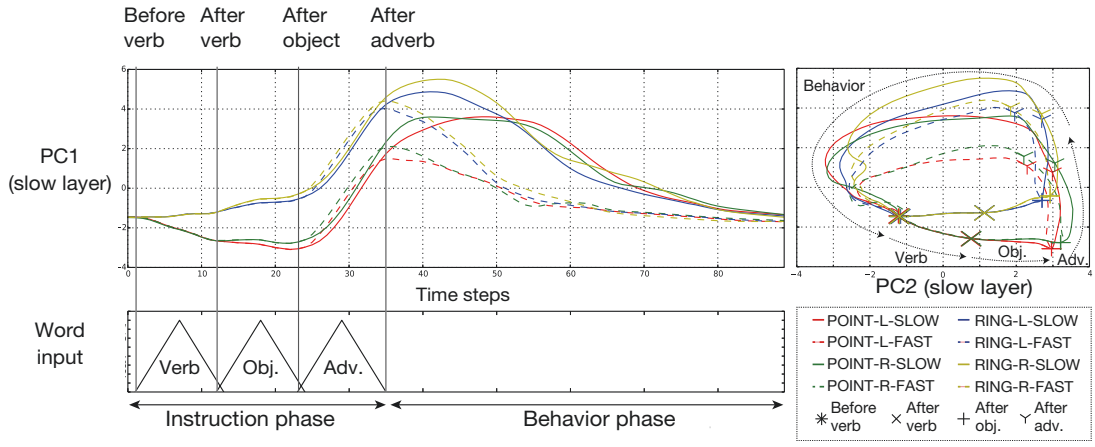

Figure 4.6. [Top-left] The time development of the context state of the slow layer in the PC1 direction. The contribution ratio was 28.3%. Respective lines show the representative development of one of eight different behaviors. The context state develops along one branch in accordance with the meaning of the given words. [Top-right] The same development in the PC1–PC2 subspace. This space clearly shows the cyclic attractors that directly represent the task progress pattern, which is multiple repetitions of instruction reception and behavior response.

The top-right panel of Figure 4.6 also visualizes the same time development by projecting it into the PC1–PC2 subspace and by hiding the time step axis instead. This visualization clearly shows that cyclic attractors that directly represent the task progress pattern — namely repetition of instruction reception and behavior generation — as cycles were organized in the model's forward dynamical system. After behavior generation, the context state reaches the vicinity of the initial state (indicated by an asterisk) again. We also confirmed that the initial state was organized as a fixed-point attractor, thereby enabling the robot to stably wait for a subsequent instruction. Thanks to such cyclic attractive dynamics, the model successfully continued the current interactive task. The autonomous phase-switching from instruction reception through behavior generation to stand-by was achieved by a simple series of forward propagations without any external cues.

It should be noted that when the instruction was C-type, a branch that the context state develops along differs in accordance with the current bell arrangement. For instance, "blue" means either LEFT or RIGHT depending on which side the blue bell is. Even in these contexts, the model successfully chose the correct branch by having learned the relationships of the color words and the bell colors. Here, the bifurcating structure is realized by the model's dynami-

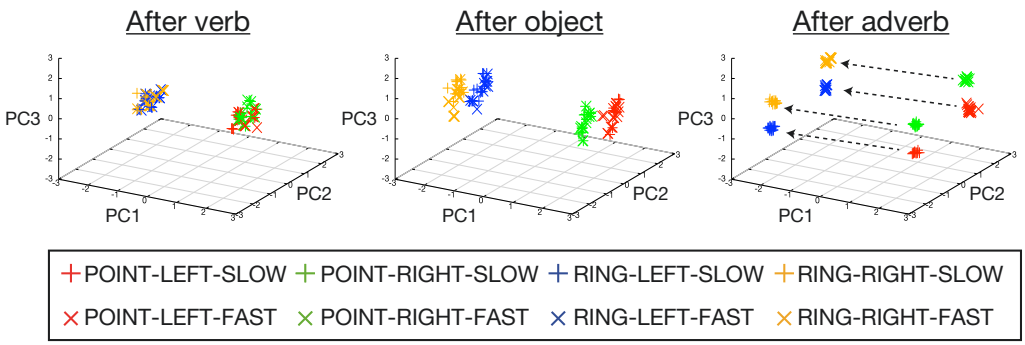

Figure 4.7. [From left to right] The context states of the slow layer after receiving a verb, object, and adverb, respectively, in the PC1–PC3 subspace (their respective contribution ratios were 52.2, 40.0, 8.5%, and we extracted PCs only from the states just after the instruction phase). Each plot corresponds to a different episode. The panels show the doubling of clusters according to the given words. The dashed arrows in the right panel roughly illustrate the vectors connecting representative grounding points for two behaviors that differ in only the verb element (for instance, POINT-RIGHT-SLOW and RING-RIGHT-SLOW), namely the POINT–RING axis. The cosine similarities between two out of four vectors in the original space were 0.972 at minimum, in other words, they were all almost parallel. In the cases of the LEFT-RIGHT and SLOW-FAST axes, the minimum cosine similarities were 0.938 and 0.977, respectively.

cal system. Therefore, the state development is not completely identical among different episodes, having variance from the average development. This variance is caused by the visual fluctuation or the influence of previous episodes. This fact is shown in Figure 4.7. The panels in Figure 4.7 show the context states after receiving a verb, object, and adverb by projecting them onto the PC1–PC3 space. Each plot corresponds to a different episode. The point colors and shapes differ in accordance with the behavior to be executed in each episode. The panels show that each word input leads to a doubling of clusters. Finally, after adverbs, the eight clusters corresponding to the eight behaviors appear. In this way, the sentences are grounded in robot behaviors as cluster structures in the middle of cyclic attractive dynamics.

Furthermore, we can see that the topology of these clusters reflects the semantic structure of given sentences. More precisely, Each of these clusters was arranged on a vertex of a parallelepiped, whose axes corresponded to POINT–RING, LEFT–RIGHT, and SLOW–FAST pairs. In the rightmost panel of Figure 4.7, the four dashed arrows roughly illustrate the vectors connecting representative grounding points for two behaviors that differ in only the verb element

(e.g., POINT-RIGHT-SLOW and RING-RIGHT-SLOW), namely the POINT–RING axis. The cosine similarities between two out of four vectors in the original space were 0.972 at minimum, in other words, they were all almost parallel. In the cases of the LEFT-RIGHT and SLOW-FAST axes, the minimum cosine similarities were 0.938 and 0.977, respectively. On the other hand, the cosine similarities between the POINT-RING and LEFT-RIGHT axes, between the LEFT–RIGHT and SLOW–FAST axes, and between the SLOW–FAST and the POINT–RING axes were 0.079, 0.035, and 0.075, respectively, meaning that these axes were close to orthogonal. By non-linearly encoding the input words, which were orthogonal to each other, the model organized this parallelepiped representation space.

Taken together, the slow layer learned to self-organize the dynamical system that worked synchronously with the progress of the current interactive task. The instruction reception, behavior generation, and stand-by phases were represented as different parts of the cyclic attractors. The grounding of linguistic instructions in behavioral sequences was encoded as a topologically organized cluster structure that was developed by bifurcations in accordance with the input words.

4.5.2 Fast Layer Dynamics

We also applied PCA to the internal states of the fast layer to visualize its self-organizing dynamics. Figure 4.8 shows that the fast layer does not maintain memory in the long term, rather its state flexibly changes in accordance with the current input and output. While receiving a verb, the context state of the fast layer develops differently in accordance with the given verb. After shifting into the object input phase, the context state immediately forgets the information about the verb. Instead, the context state develops in accordance with the currently given object. The same is true in the case of a shift from the object phase to the adverb phase. Instead, the information vanished from the fast layer is fed into the slow layer and maintained as described in the previous subsection. On the other hand, during behavior generation, the information maintained in the slow layer is fed back to the fast layer. Getting information from the slow layer, the context state of the fast layer develops differently in accordance with the behavior to be

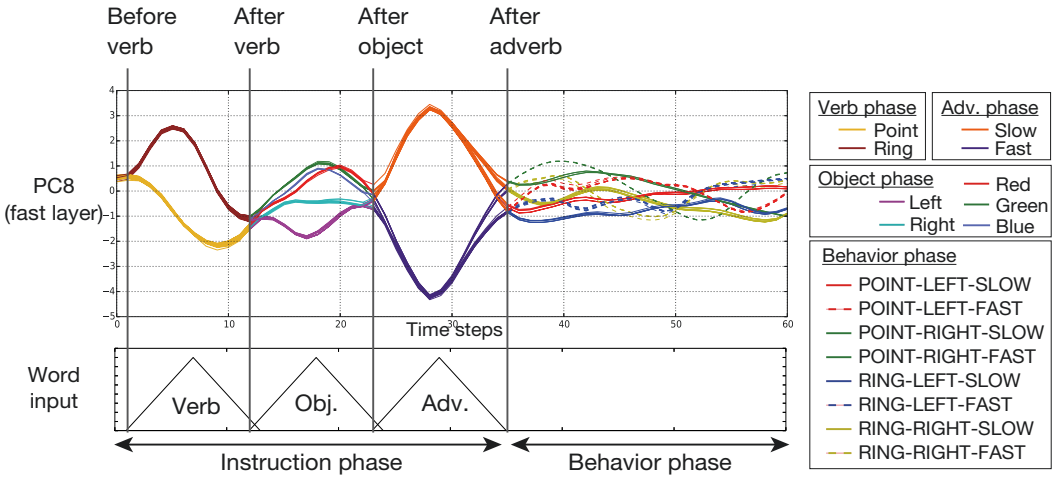

Figure 4.8. The time development of the context state of the fast layer in the PC8 direction (the contribution ratio was 3.0%). Each line shows the development during one episode. The lines are colored differently in the verb, object, adverb, and behavior phases. The context state of the fast layer develops very differently in accordance with the current I/O flow; it immediately forgets past information.

executed[8].

4.5.3 Comprehensive Summary of the Model Mechanism

We describe the mechanism of the whole of the model by considering all analysis results together. The global representation of the context in task progress was organized in the slow layer in the form of cyclic attractors. The grounding of language in behavior was represented as relay points in the middle of the cyclic attractive dynamics. However, the actual multimodal I/O flows are more complicated than the global representation by the slow layer. Thus, the fast layer, whose state changes more drastically, transforms between the current I/O states and the slow layer representations. While receiving an instruction, the fast layer integrates the word input with visual input and feeds the integrated information to the slow layer. By receiving such information, the slow layer reaches an appropriate activation in accordance with the behavior to be executed. This is a

[8]The reason why we plotted the time development in the PC8 direction is that this component included the information about each part of speech and joints relatively evenly, meaning this fact could be seen readily. Although other principal components had a similar tendency, they simultaneously tended to primarily correspond to a specific modality. For instance, PC1-4 were primarily used to represent the current joint angle values.

bottom-up mechanism. On the other hand, during behavior generation, the model works in a top-down manner. The transition along the second half of the cyclic attractor is transformed into a detailed joint angle trajectory thanks to a non-linear transformation through the high-dimensional space of the fast layer. By hierarchically self-organizing the functions that work on different timescales on its different layers, the model allowed the robot to continue to execute the current interactive grounding task.

4.6 Discussion

In this chapter, we performed a robot experiment to evaluate whether the proposed method can learn the conversion task from language to robot behavior depending on the visual context. The model could continue the given interactive task by learning to ground linguistic instructions in robot behavioral sequences conditioned on the current visual context and by utilizing the acquired relationships in appropriate interaction contexts.

During task execution, the latent representation of grounding was achieved by bifurcating in accordance with the linguistic command and the visual context. From the grounding point, the model produced an appropriate behavioral sequence in the subsequent forward propagation, while the context state moves along the second half of the attractor. Furthermore, by organizing a fixed-point attractor at the initial point, the model could wait for a subsequent instruction stably in the stand-by posture. Thanks to this structure, which represented both grounding relations and the task progress pattern, the robot successfully continued the current interactive task by autonomously switching between instruction reception, behavior generation, and stand-by phases.

In the following subsections, we discuss the characteristics of the current model in comparison with other existing grounding methods and indicate its advantages and disadvantages.

4.6.1 Encoding Grounding Structure as Fixed-dimensional Representations

Encoding grounding as fixed-dimensional latent representations is an effective way to cope with the "linear nature" of language indicated by Saussure [97]. Natural language expressions are restricted in the sense that they are able to express their referent only as a linear series of words that correspond to certain features of the referent and that compositionally build the whole meaning, whereas the word order is uncorrelated to the actual temporal structure of the referent. In the current task, the RING-LEFT-FAST behavioral sequence is not able to be temporally reduced to primitives corresponding to "ring," "left," and "fast." These words express a certain feature that relates to the whole of the behavioral sequence and compose the meaning by being arranged in accordance with syntactic rules. The combinatorial structure of language is intrinsically different from that of the real world.

Considering this restrictive fact that a language system has combinatorial structure different from behavior and that both still have a temporal context, a method that learns to embed their relationships in a fixed-dimensional latent representation can achieve their grounding in a unified manner. In the current task, instruction understanding was represented as a branching structure that developed a cluster structure corresponding to the behavior to be executed (from language to behavior). Heinrich and Wermter [98] performed a robot experiment in which an MTRNN learned to generate various descriptions of robot vision and proprioception sequences through the fixed-dimensional representations (from behavior to language). Ogata et al. [41] showed that two RNNs compounded by a PB module, which has a small number of neural nodes to connect two RNNs, could bidirectionally convert between linguistic expressions and robot behavioral sequences through the fixed-dimensional PB space (both directions). As in these cases, grounding that consists of recognition and generation of variable-length sequences of language and behavior can be accomplised through fixed-dimensional representations by RNNs. This structure is totally learned from data without manually designed encoding and decoding rules.

4.6.2 Topologically Organized Representation Space

The analysis of the fixed-dimensional latent representations in the slow layer revealed that the grounding of language in behavior was represented in a topologically organized cluster structure. This sort of structure, which reflects the compositional nature of language, was also seen in a work by Sugita and Tani [9]. To ground two-word instructions in a set of robot behaviors, they employed two RNN modules — one dealt with language and the other with behavior — and a PB module. Training that constrained PB values for generating a behavior and its paired sentence to be close to each other resulted in a topological structure that represented meanings of words and their compositionality in the PB space. In this framework, to search an optimal PB vector (grounding point) to convert an instruction to a behavioral response, the repetitive back-propagation and updates of PB values were required. In contrast, our experiment in this chapter demonstrated that a similar structure for representing language grounding can be achieved by the forward-path only. This is an advantage because using only forward-propagation requires less calculation. Moreover, a topologically organized structure has the potential of generalization to novel sentences. We must evaluate such a generalization capability; this is more difficult than the combinatorial generalization between instructions and visual input demonstrated in the current experiment.

A topologically organized structure in NN's state space has also be reported in the field of NLP. Mikolov et al. [99] demonstrated that training an RNN using a language modeling task with a large corpus resulted in effective distributed representations of word meanings. In the embedding space, some algebraic operations could be performed (e.g., "Madrid" − "Spain" + "France" = "Paris"). It was also reported that such an algebraic operation can be performed on representation spaces of vision [100, 101]. For example, an image of a woman with sun-glasses can be generated by a manipulation of [man with sun-glasses] −[man without sun-glasses] + [woman without sun-glass] in a latent space. If it is revealed in future that such algebraic operations can also be performed on the representation space of grounding, it will be of much practical use.

4.6.3 Characteristics of F-EDM Compared to those of S-EDM

This chapter performed an evaluation experiment for an F-EDM that has no explicit separation of the instruction reception and behavior generation phases. This model is trained with predictive learning where it constantly receives current word, vision, and joint input and generates joint angles for the future time step. In this learning, the model learns not only the relationships between language and behavior but also task progress patterns implicitly. In the case of the normal EDM, i.e., S-EDM, the phase switch of recognition and generation is externally given. However, in the case of F-EDM, the model after training can execute the task by autonomously switching between instruction reception, behavior generation, and stand-by phases just in forward calculation. In the current task, the task progress pattern is relatively simple, so the benefits of F-EDM may be not so abundant. However, in the case that the contextual information in previous episodes is required to achieve the current episode or that an additional instruction is given during behavior generation, the F-EDM would bring about great benefits. In such cases in which the required grounding is not presented as simple successive pairs of instruction and behavior, to manually design task progress patterns becomes difficult. F-EDMs have the potential to flexibly deal with such situations by implicitly learning the various task progress patterns if experiences include these situations. In future work, we must evaluate whether F-EDMs actually learn such tasks.

4.6.4 Stability and Safety of Model Behavior

We must also consider the stability of the acquired dynamics. In the current experiment, the model successfully responded to instructions by generalization, even in the unexperienced cases. We also confirmed that the stand-by state to wait for an instruction was represented as a fixed-point attractor that was stable to a certain extent. However, we cannot ensure that the model's working is globally stable even in unexpected situations or in very noisy environments. It is extremely difficult to globally analyze the characteristics of a high-dimensional dynamical

system. The current framework works by executing the forward-propagation only; thus, there would be a risk that the model's behavior suddenly becomes unstable when an unexpected event occurs. To ensure safety for practical use, some protective module that controls output joint torques or that monitors error values should be implemented.

4.6.5 Scalability of the Proposed Method

We must also evaluate the model's scalability. Recently, existing studies have performed experiments in which S-EDMs with deep architecture were trained on natural-language-instructed navigation tasks [15, 16, 17] that imposed an agent to navigate in a virtual environment in a highly context-dependent manner. Most studies trained the models in a reinforcement manner, and the output form is often discretized, such as STRAIGHT, TURN-LEFT, TURN-RIGHT etc. Although existing studies have these differences from the current study, they reported that the models could understand relatively long instruction sentences. We must verify whether F-EDMs can also achieve similar levels of grounding capability and whether both types of EDMs successfully scale up to real-world problems, which include more noise and fluctuation than simulated problems and require the robot to execute behavior in a continuous space.

4.7 Summary

In this chapter, we evaluated our proposed framework on **the context dependency of meanings of language (R1)**. We designed a task that required the model to unidirectionally convert language instructions into robot behavioral responses by considering the current visual context. The experiment demonstrated that the model organized the forward dynamics that represented the task progress patterns from learning in an end-to-end manner. Also, the model embedded the context-dependent relationships between language and behavior in the middle of the forward dynamics by encoding visual information together with word input. Thanks to such a structure, the model successfully generated robot behaviors appropriate for the visual context in response to given instructions. Figure 4.9

summarizes the topics discussed in this chapter.

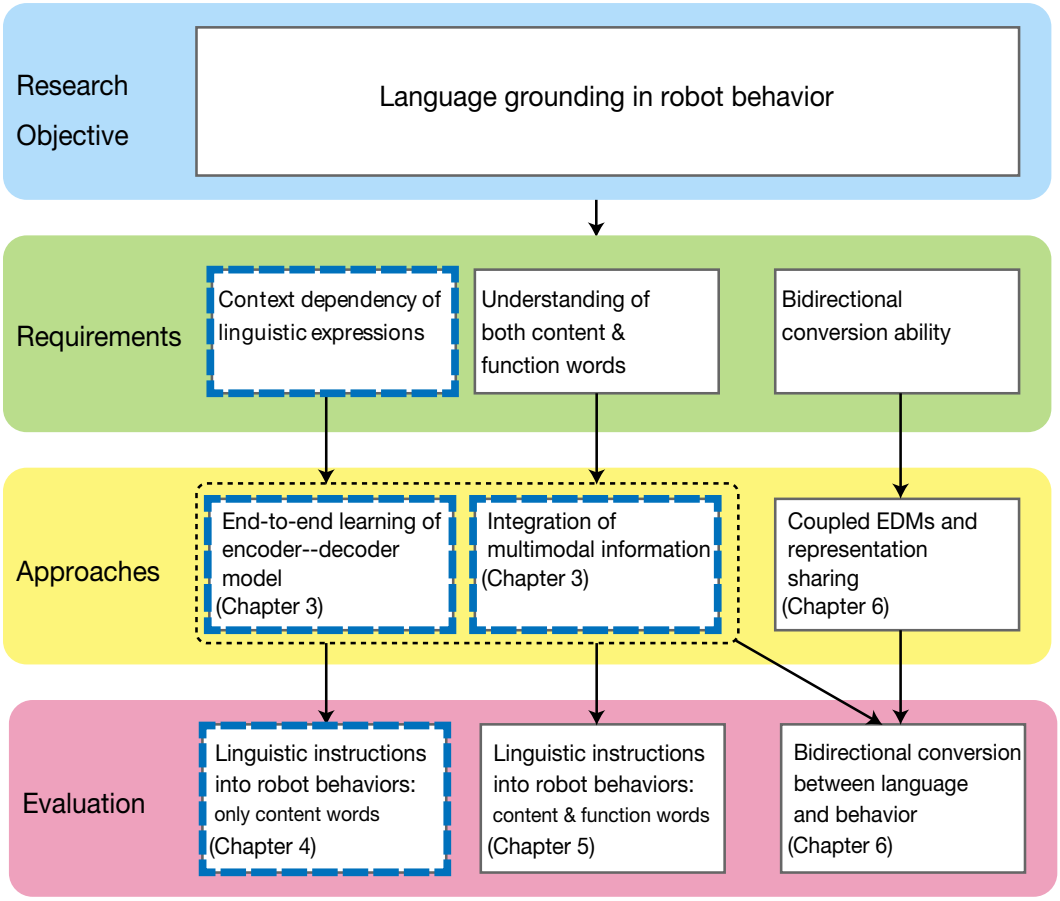

Figure 4.9. The main targets discussed in Chapter 4.

Chapter 5

Understanding Both Content Words and Function Words

5.1 Introduction

The previous chapter performed an evaluation experiment and confirmed that the F-EDM could learn to convert context-dependent instructions into robot behavioral responses. In the experiment, the meanings of instruction sentences depended on visual context but the sentences included only content words. Next, this chapter evaluates our proposed framework on the **understanding of both content and function words (R2)**. Specifically, as an example of function words, we perform an experiment with some logic words, e.g., "not", "and", and "or". Although logic words do not have direct grounding in the real-world matters, they contribute to the construction of meanings of natural language expressions by working as logical operators. For instance, immediately after someone has opened a window, the instructions "close the window" and "don't open the window" would be requests to produce the same behavior of CLOSE-WINDOW. As another example, an instruction "pick the box or the ball" would accept both PICK-BOX and PICK-BALL as appropriate responses.

In the field of cognitive robotics, existing studies on integrative learning of language and behavior have rarely focused on such logic words. Probabilistic models generally learn the stochastic co-occurence relations between content words and multimodal features, so they have not investigated how function words combine

the meanings of content words [40, 55]. NN models have the potential to encode instructions that consist of both content and function words as distributed representations. However, previous studies using NN models have also mainly dealt with grounding learning of sentences that consisted of content words only [9, 41, 10]. In this chapter, we demonstrate that the proposed RNN-based EDM actually has the capability of learning to convert linguistic instructions consisting of both content words and logic words into behavioral sequences. We also investigate how the model internally represents these function words together with other content words.

The remainder of this chapter is organized as follows. In Section 5.2, we explain the detailed architecture of the model employed. In Section 5.3, we describe the experimental design. In Section 5.4, we report the task performance. Section 5.5 performs analyses of how the model internally represents both grounded words and function words. Section 5.6 also investigates the internal representation of logic words in more detail by performing an additional learning experiment. In Section 5.7, we discuss the acquired results. Section 5.8 concludes this chapter.

5.2 Learning Model Details: F-EDM Based on LSTM-RNN

Similar to the experiment in Chapter 4, we formulate the conversion task from language instructions to behavioral sequences as a problem to learn to produce a robot's joint angle trajectory appropriate for the current context. Therefore, we employ the F-EDM. However, we use an LSTM-RNN instead of an MTRNN. Recently, a great deal of studies have reported that RNNs with LSTM units have strong capability to deal with temporal patterns of data. LSTM units have a cell to memorize contextual information and three types of gates that explicitly control information flow that goes in and out of the cell. Input gates control the information to be drawn into the cell. Forget gates determine whether the current cell state can be refreshed or not. Output gates determine whether the current state should be output or not. By introducing these learnable gates, LSTM-RNNs can memorize much longer temporal patterns and also ease the vanishing

gradient problem during training. For details of the cell update calculation, refer to Appendix A.3. Moreover, in the case of MTRNNs, we must tune the time constant in each layer as a hyperparameter. In contrast, the LSTM does not require it.

Although the task learned in this chapter is actually not required to memorize extremely long temporal structure, it still requires the model to solve complex problems such as grounding perfectly orthogonal sentences in the same behavior. Utilization of an LSTM-RNN that can flexibly embed input stream thanks to its high non-linearity is an advantageous choice to address the current task[1].

5.3 Experimental Setup

5.3.1 Task Design

This section describes the task design for the learning experiment. Although we completely perform the experiment as a numerical simulation on a computer, we describe the task as though a robot executed it for readers to be able to understand intuitively. First, the robot holds two out of three flags whose colors are red, green, and blue; it holds one with the left arm and the other with the right arm. The robot then receives a multi-word instruction. The instruction consists of a truth value ("do", "don't"), a verb ("raise", "lower"), and an object ("red", "green", and "blue") in reverse order[2]. An object word refers to an arm holding the flag of the stated color. A verb specifies if the robot must lift or lower the flag. Eventually, "do" directs the robot to behave by following the verb. In contrast, "don't" directs the robot to produce the inverse behavior. For instance, when the robot is holding the blue flag in its left arm receives the instruction "don't raise blue," the robot must lower the left arm (LEFT-DOWN)[3].

[1] In fact, we evaluated the performance of MTRNNs with some hyperparameter sets on the current task as a preliminary experiment. However, they could not achieve learning.

[2] We give the words in reverse order to common English because we designed the current task by extending a famous game in Japan and Japanese is an SOV language.

[3] In this task, "not" is always given as a form of "don't." We treat each of "do" and "don't" as one word as if they were paired truth values. On the other hand, in the additional experiment in Section 5.6, we design another task in which "not" is given as an independent word.

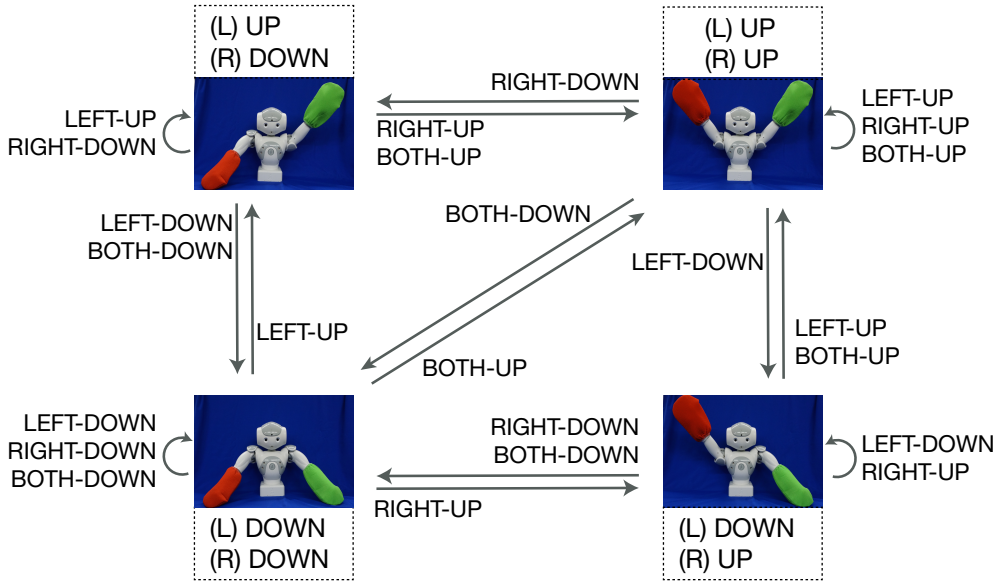

Figure 5.1. The possible transitions among the robot postures in the flag task.

We also make instructions that concatenate two object words by "and." We refer to them as AND-concat instructions. For instance, when the robot is holding blue and green flags and receives the sentence "do lower blue and green," both arms must be lowered. Furthermore, we make instructions that concatenate color words by "or." We refer to them as OR-concat instructions. For instance, when the robot is holding the green and red flags and receives the sentence "do raise red or green," the appropriate behavior is lifting up either arm. Thus, the current task includes six goal-oriented behavioral patterns: LEFT-UP, LEFT-DOWN, RIGHT-UP, RIGHT-DOWN, BOTH-UP, and BOTH-DOWN. Note that even if the same goal-oriented behavior is directed, the actual joint angle trajectory that the robot must generate varies in accordance with the current arm posture (illustrated in Figure 5.1). There are even situations in which neither arm moves.

Taken together, this task includes all combinations of flag colors (6 patterns), robot's own postures (4 patterns), and language commands (24 patterns), which means there are 576 ($6 \times 4 \times 24$) possible situations. We never give sentences that are inconsistent with the flag colors. For instance, when blue and green flags are held, we never give the sentence "red up true." Moreover, two grasped flags always have different colors from each other.

This task imposes the following requirements on the model. First, an arm that an object word indicates depends on which arm the flag of the stated color is held. The task requires grounding in a current visual context similar to the task in the previous chapter. Secondly, an actual motion trajectory that the robot must generates also depends on the robot's own arm posture. For instance, when an instruction directs the robot to produce RIGHT-UP behavior, if the right arm is currently in the DOWN posture, it must be lifted up. In contrast, if the right arm is currently in the UP posture, the robot must maintain the UP posture. Finally, the model must cope with both content words (e.g., verbs and objects) and logic words, namely "do", "don't", "and", and "or". The logic words are examples of function words that we concentrate on in this experiment. Due to the design of the current task, in extreme situations, instructions completely orthogonal to each other refer to the same behavior (for instance, "don't lower green" with the green flag in the right hand and "do raise red" with the red flag in the right hand). On the other hand, OR-concat instructions are ambiguous because they accept multiple types of behavior as correct even in the perfectly same context.

5.3.2 Target Data

The current task is represented as a succession of 14-dimensional vectors. Table 5.1 describes what each element represents. We assign nine dimensions for words. Instructions are given as a series of one-hot vectors. We assign three elements to vision. These three elements are assigned to the red, green, and blue channels. When the color is held in the left arm, the element takes a value of 0.8; when the right arm holds it, the element takes a value of -0.8; and if neither arm holds it, the element is set to 0.0. We use only the shoulder pitch on both arms, so we assign two elements for proprioception. The UP and DOWN postures correspond to values of 0.8 and -0.8, respectively. Posture changes from DOWN (UP) to UP (DOWN) after receiving a sentence are executed over 6 time steps. Figure 5.2 shows a data example. We added a small amount of Gaussian noise to the joint angle values[4].

[4]In the preliminary experiment, the model learned without noise obtained much poorer results. By adding noise, we could improve the resulting performance. This effect is discussed

Table 5.1. Data representation of the flag up–down task.

Elements	Descriptions
w_0	"do"
w_1	"don't"
w_2	"raise"
w_3	"lower"
w_4	"red"
w_5	"green"
w_6	"blue"
w_7	"and"
w_8	"or"
v_r	With which arm the red flag is grasped
v_g	With which arm the green flag is grasped
v_b	With which arm the blue flag is grasped
j_l	The shoulder pitch of the left arm
j_r	The shoulder pitch of the right arm

5.4 Learning Results

5.4.1 Evaluation Method and Model Hyperparameters

We made a training dataset that consists of 2048 sequences, each including ten episodes. We randomly ordered the situation patterns in each data sequence. The dataset included all the 576 possible situation patterns at least once. We constructed five models whose recurrent cell is a 1-layer LSTM with 50, 70, 100, 150, and 300 nodes. For each model, we performed training ten times with different random seeds. Moreover, we also trained the 100-node network using data to which Gaussian noise was not added on the joint angle values. As an optimizer, we used the Adam [82] algorithm with a learning rate of 0.001. The parameter update was performed 10,000 times.

After training, we made a test dataset. In the test dataset, all the possible situation patterns were included ten times. The episode order was random and different from the training dataset. When the absolute distance between the predicted joint angle on both arms six steps after the command reception and

in Section 5.7.

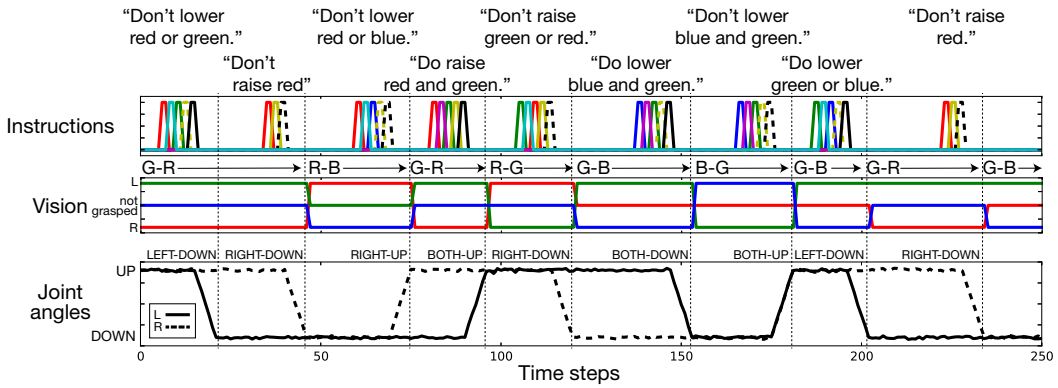

Figure 5.2. An example target of the flag task. Vertical dashed lines indicate ends of episodes. [Top] We give sentences as sequences of multiple words. The words are each represented in the one-hot form. During stand-by and behavior production, zero vectors are given. [Middle] Vision is given as a succession of three dimensional (red, green, and blue) vectors. The colors of flags randomly change after every behavior phase. We perform this experiment as a numerical simulation on a computer, so the flag colors instantaneously change. The grasped flags occasionally do not change (e.g., the case between the first and second episodes in this figure). [Bottom] A behavior execution follows a command reception immediately.

the target one was smaller than 0.04, we judged that the model had successfully generated appropriate behavior. Here, some OR-concat instructions allow two kinds of behaviors as appropriate. In these situations, if the model has successfully generated one of the appropriate behaviors, we judged the episode as a success. We regarded the situational patterns in which the model successfully generated appropriate behavior at least seven times out of ten trails as "successfully learned."

5.4.2 Task Performance

We classified situation patterns into four types. (1) Situations that give a sentence with one color word (192 patterns). (2) Situations that give an AND-concat instruction (192 patterns). (3) Situations giving an OR-concat instruction that allows only one correct behavior (144 patterns). For instance, when the model whose arms are both already in the UP posture receives the sentence "do raise blue or green," the only correct answer is to keep the current UP-UP posture. (4) Situations giving an OR-concat instruction that allows two correct behaviors (48 patterns). We evaluated the resulting performances by counting the number of situation patterns that the models learned successfully with respect to each of the

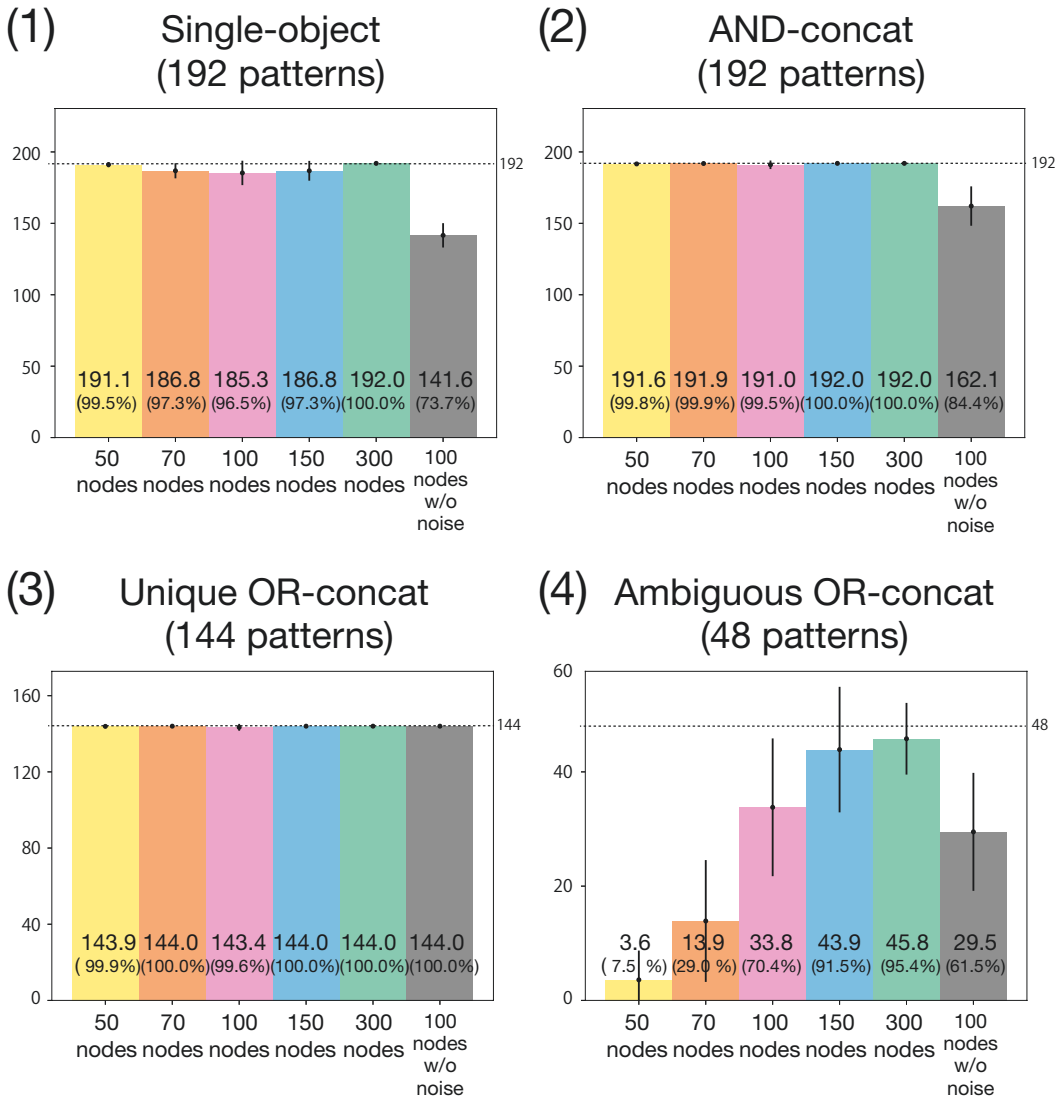

Figure 5.3. The results of behavior generation. We scored the resulting performance by counting the number of situation patterns the models successfully learned with respect to each of the four situation types: (1) single-object instructions, (2) AND-concat instructions, (3) OR-concat instructions, which allow only one correct behavior, and (4) OR-concat instructions, which allow two correct behaviors. The annotated scores are mean values of 10 trials with different random seeds. The error bars represent standard deviations.

four situation types.

We illustrate the performance results in Figure 5.3. The score on each bar is the mean of ten trials with different random seeds. All the models successfully learned most of the situations of types (1), (2), and (3), which have only one correct behavior. The only exception was the 100-node model that learned with

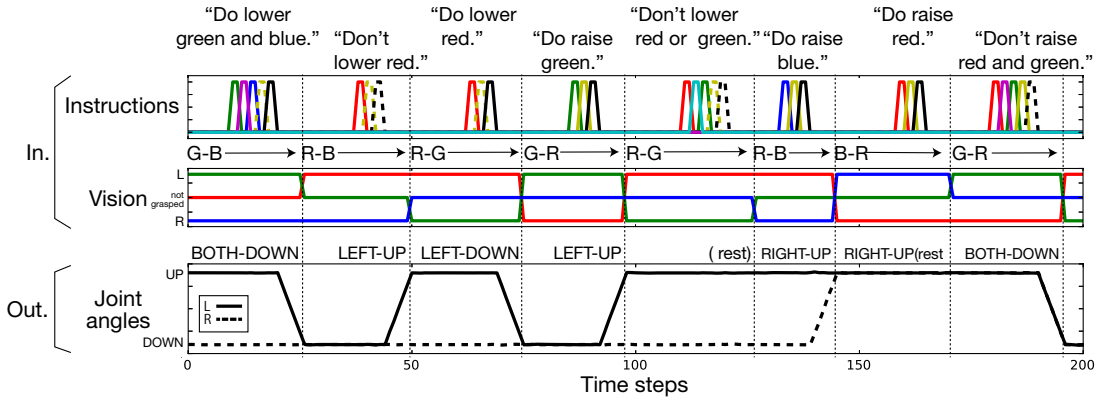

Figure 5.4. An example of the resulting interaction by the 300-node model. It successfully generated appropriate behavior in almost all episodes.

noiseless joint angle data. This model was not able to learn these types sufficiently well. In situations of type (4), which allow multiple correct behaviors, a clear difference was seen among models: increase in the number of LSTM nodes improved performance. Figure 5.4 shows an example generated by the 300-node model. The model successfully generated appropriate behavior immediately after every instruction reception.

After that, we counted which arm the model actually moved in the type (4) situations. If the models learned to respond to ambiguous OR-concat instructions just as left arm behavior or just as right arm behavior, which still fulfill the aforementioned evaluation criteria, we cannot regard the meaning of "or" as being truly learned. We checked the result by the 300-node model. In 52.5% of the test trials, the model moved the right arm. In 45.4%, the model moved the left arm. In 2.1%, neither correct behavior was successfully executed. Overall, the model moved both arms relatively evenly in the type (4) situations. This task includes 48 patterns of type (4) situations, and we carried out tests ten times for each of these patterns. For all patterns, the model sometimes moved the right arm and other times moved the left arm. These results show that the model learned the meaning of OR-concat sentences appropriately as "or." Taken together, the models successfully learned to perform the current task.

5.4.3 Generalization Capability

The previous subsection reported that the model trained with all the possible situations could behave appropriately in most cases. To investigate the model's generalization capability, this subsection trained the models again using only 50% or 75% of the possible situation patterns. Situations included in the training dataset were regularly chosen as shown in Table 5.2. Therefore, each word, flag order, and arm posture would be experienced uniformly. We trained 100-, 150- and 300-node models only. Figure 5.5 shows the resulting performances.

First, we describe the results of the models that learned with 50% of the possible situation patterns. In many of the type (2)–(4) unexperienced situations, the models successfully generated appropriate behavior. On the other hand, the models successfully dealt with only approximately one-third of the type (1) situations with a single-object instruction. This score is almost the same as random chance. To clarify why the models failed to execute behaviors appropriately, we sampled some failure episodes executed by the 100-node network (Figure 5.6). In one failed case (enclosed by the yellow box), although the arm reached the goal posture, the generated trajectory was unstable. Therefore, the criterion that the absolute distance between prediction and target must be less than 0.04 was not satisfied. In the other failed case (blue box), the model generated a wrong behavior. The second failure suggests that although the model roughly understood to execute some possible behavior after receiving an instruction, it had not attained comprehension of the relations between object words and visually given flag colors.

One possible reason that the models failed to behave appropriately in response to one-object instructions is that only this type of instruction truly depends on visual context. For instance, when an AND-concat instruction is given, the model has no need to take visual context into account. The reason is because whenever a sentence is AND-concat, the model has to move both arms, regardless of the colors of the held flags. In fact, when we gave the model holding the blue and red flags the instruction "do raise blue and green" that was contradictory to the flag colors as a trial, the model lifted both arms. This suggests that the model ignores

Table 5.2. To investigate the model's generalization capability, we performed training again using only (a) 50% or (b) 75% of the possible situations. (a) In the former case, only the situations indicated by double check marks were included in the training data. (b) In the latter case, situations indicated not only by double-check marks but also by single-check marks were included in the training data. The situations denoted as an empty cell were not included in the training data in either case. In this table, instruction patterns are abbreviated as follows. T: do; F: don't; U: raise (up); D: lower (down); L: left flag color; R: right flag color; A: AND-concat objects; O: OR-concat objects. For instance, the cell referred to as DOWN-DOWN, R-G, TUR is indicated by a double-check mark. This means that it is possible that the robot waiting in a DOWN-DOWN posture and grasping R-G flags receives an instruction "do raise green" during training in both cases of (a) and (b). As another example, the cell referred to as UP-UP, R-B, TUO is indicated by a single-check mark. This means that it is possible that the robot waiting in an UP-UP posture and grasping R-B flags receives an instruction "do raise red or blue" and "do raise blue or red" during training in only the case of (b). As another example, the cell referred to as DOWN-UP, B-R, TUL is denoted as empty. This means that the robot grasping B-R flags and waiting in an DOWN-UP posture does not receive an instruction "do raise blue" during training in either case.

Posture	Colors	Instructions															
		TUL	FUL	TDL	FDL	TUR	FUR	TDR	FDR	TUA	FUA	TDA	FDA	TUO	FUO	TDO	FDO
DOWN-DOWN	R-G	✓	✓✓		✓✓	✓✓		✓✓	✓	✓		✓✓	✓✓	✓✓		✓✓	✓
	G-R	✓✓		✓✓	✓		✓✓	✓	✓✓	✓✓	✓	✓✓			✓✓	✓	✓✓
	G-B	✓	✓✓	✓✓		✓	✓✓	✓✓			✓✓	✓	✓✓	✓✓	✓✓		✓
	B-G	✓✓	✓		✓✓	✓✓	✓		✓✓	✓	✓✓	✓✓		✓		✓✓	✓✓
	B-R		✓	✓✓	✓✓	✓✓	✓✓		✓	✓✓	✓		✓✓	✓✓	✓	✓✓	
	R-B	✓✓	✓✓	✓		✓		✓✓	✓✓	✓✓	✓✓		✓		✓✓	✓	✓✓
DOWN-UP	R-G		✓	✓✓	✓✓	✓✓	✓✓	✓		✓✓		✓	✓✓	✓✓	✓	✓✓	
	G-R	✓✓	✓✓		✓		✓	✓✓	✓✓	✓✓	✓✓		✓	✓		✓✓	✓✓
	G-B	✓	✓✓		✓✓	✓✓		✓✓	✓	✓		✓✓	✓✓	✓✓	✓✓		✓
	B-G	✓✓	✓	✓✓		✓	✓✓		✓✓	✓✓	✓	✓✓		✓✓	✓		✓✓
	B-R		✓✓	✓✓	✓		✓✓	✓✓	✓		✓✓	✓	✓✓		✓✓	✓✓	✓
	R-B	✓✓		✓	✓✓	✓✓	✓		✓✓		✓✓	✓✓	✓	✓	✓✓		✓✓
UP-DOWN	R-G	✓	✓✓	✓✓		✓	✓✓	✓✓			✓✓	✓	✓✓	✓	✓✓	✓✓	
	G-R	✓✓		✓	✓✓	✓✓		✓	✓✓	✓	✓✓	✓✓			✓✓	✓	✓✓
	G-B	✓		✓✓	✓✓	✓✓	✓✓		✓	✓✓	✓		✓✓	✓✓		✓✓	✓
	B-G	✓✓	✓✓		✓		✓	✓✓	✓✓	✓✓	✓✓		✓		✓	✓✓	✓✓
	B-R		✓✓	✓	✓✓	✓✓	✓	✓✓		✓		✓✓	✓✓	✓✓	✓✓	✓	
	R-B	✓✓	✓	✓✓			✓✓	✓	✓✓	✓✓		✓✓	✓	✓✓	✓		✓✓
UP-UP	R-G	✓✓	✓✓	✓		✓		✓✓	✓✓	✓✓	✓✓	✓		✓✓		✓✓	✓
	G-R		✓	✓✓	✓✓	✓✓	✓✓	✓		✓✓	✓		✓✓		✓✓	✓	✓✓
	G-B	✓✓		✓	✓✓	✓✓	✓		✓✓		✓✓	✓✓	✓	✓		✓✓	✓✓
	B-G		✓✓	✓✓	✓	✓	✓✓	✓✓			✓✓	✓	✓✓	✓✓	✓✓	✓	
	B-R	✓✓		✓✓	✓		✓✓	✓	✓✓	✓✓	✓	✓✓			✓	✓✓	✓✓
	R-B	✓	✓✓		✓✓	✓✓		✓✓	✓	✓		✓✓	✓✓	✓	✓✓		✓✓

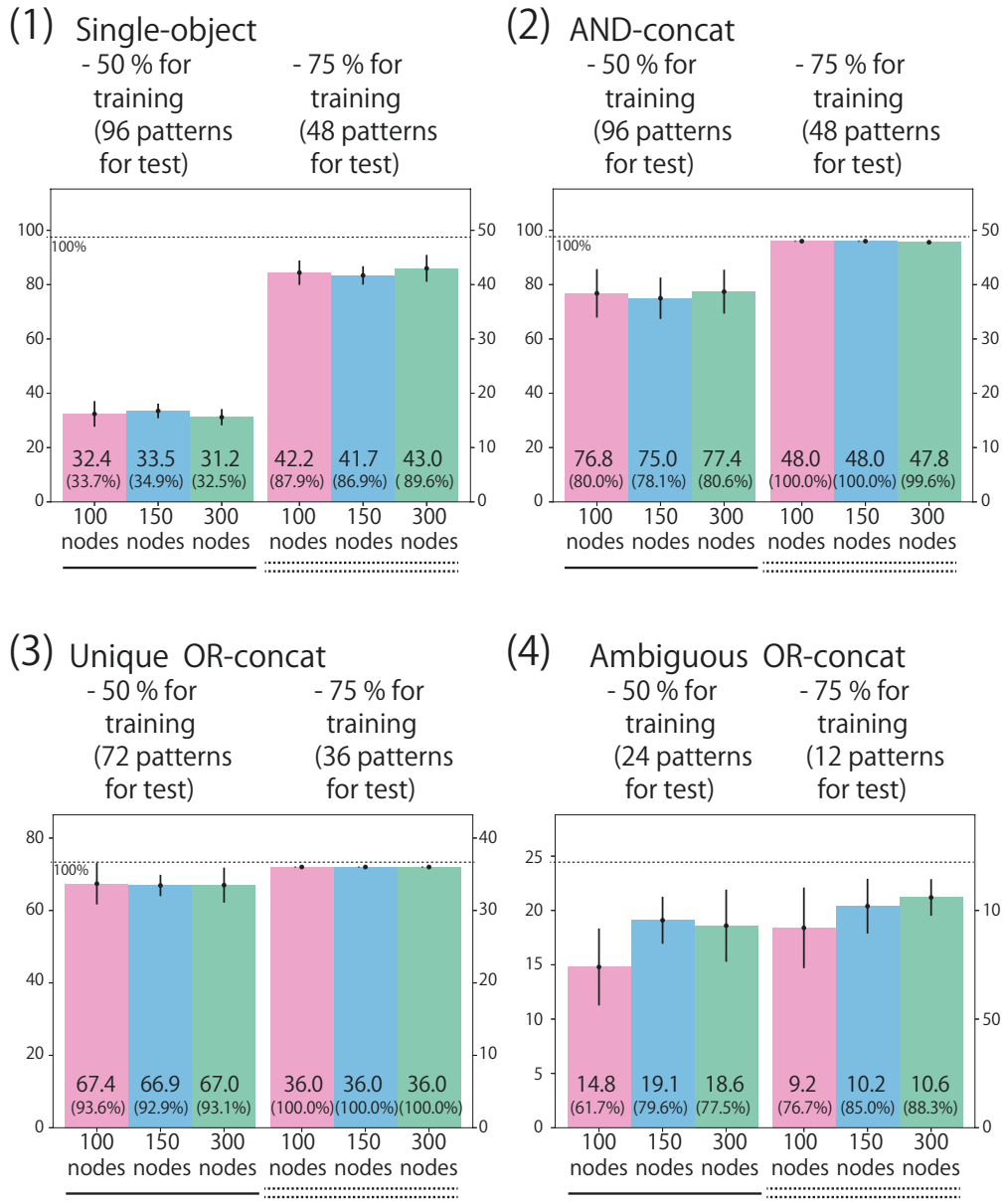

Figure 5.5. To investigate the generalization capability on the current task, we trained the models again using only (a) 50% or (b) 75% of the possible situation patterns. We scored the performances by counting the number of unexperienced situation patterns that the models successfully executed appropriate behavior for. We scored the performances with respect to each of the four situation types, similarly to Figure 5.3.

visual information in the cases of AND-concat instructions. The results were similar in other contradictory cases. In the cases of type (3) and (4) OR-concat instructions, regardless of the flag colors, the model must move only one arm. In that sense, one-object instructions are more difficult than the other types of

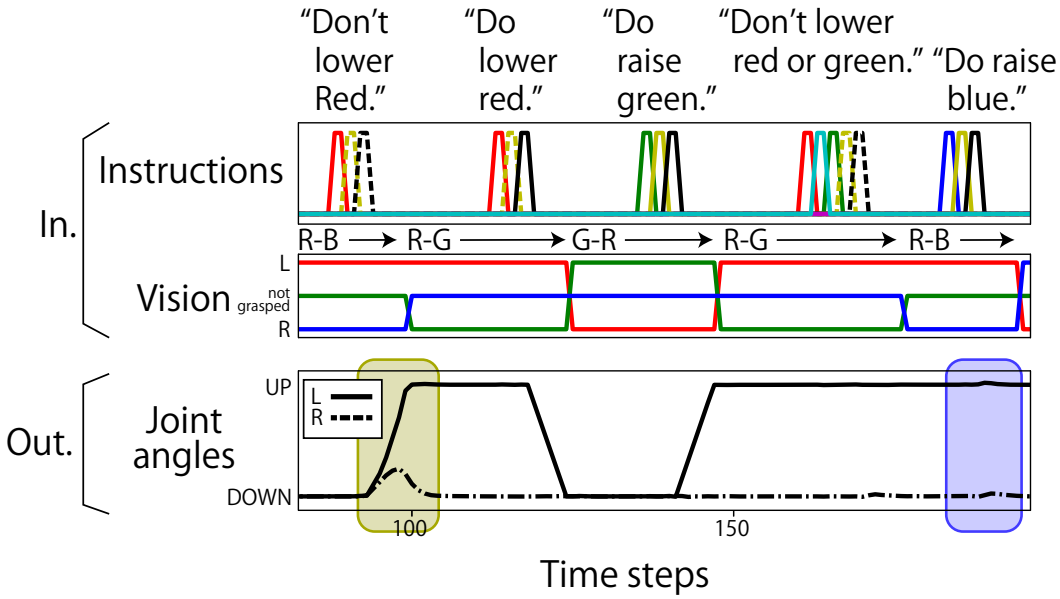

Figure 5.6. In unexperienced situations of the flag task, some different patterns of failure could be seen. [Yellow box] One failed pattern was that although the arm correctly reached the goal posture, the generated trajectory was unstable. Therefore, the model failed to satisfy the criterion that the absolute distance between prediction and target must be less than 0.04. [Blue box] The other pattern was that the model generated a wrong behavior. For instance, in the episode indicated by the blue box, the model had to lift the right arm. However, it kept the arm in the DOWN posture.

instructions. Training with 50% of the possible situations might not have been sufficient to achieve generalization covering the whole of the task space. Actually, the models that learned with 75% of the possible situations successfully dealt with more than 80% of unexperienced type (1) situations.

5.5 Analysis of Internal Representations

The previous subsection demonstrated that the F-EDM has the capability of learning to execute the flag up–down task by dealing with both content words and logic words. In the generalization test, although half of the possible patterns were insufficient, the model could generalize its experiences into most unexperienced patterns from three quarters of possible episodes. To investigate how the model internally represented the relationships between sentences, visual context, and joint angles, we applied PCA to the context states during the task execution.

5.5.1 Representations of Color Words

First, Figure 5.7 shows the context states after receiving the sentence "do raise (word indicating the left flag color)" or "do raise (word indicating the right flag color)". Here, although we fixed the robot's waiting posture to DOWN-DOWN, the situations were different depending on the colors of held flags. Thus, the model must choose which arm it lifts up by integrating the object word input and the visual context. The PC2–PC3 space directly reflects the current visual context. In contrast, PC1 represents which arm has been referred to by an object word. This suggests that by learning to predict appropriate behavior in response to instructions on the given visual contexts, the model achieved representations that corresponded to the concepts of "left" and "right".

5.5.2 Representations of Logic Words: "Do" and "Don't"

Next, we investigate how the model internally represented logic words. Figure 5.8 visualizes the context states after giving eight different single-object instructions to the model that was holding the red and blue flags in the left and right arms, respectively, and waiting in the DOWN-DOWN posture. PC1, PC2, and PC3 directly correspond to the three parts of speech, i.e., the PC3, PC1, and PC2 axes represent "do"/"don't", "raise"/"lower", and "red"/"blue" pairs, respectively. Here, the model must solve an X-OR problem caused by the "do"/"don't" and "raise"/"lower" pairs. By solving it, the model must ground its interpretation in UP or DOWN behavior. In fact, exploration of the lower-rank components revealed that the instructions that were embedded on the diagonal of the parallelogram in the PC1–PC3 subspace were embedded in the same side in the PC4 direction. In the right panel, "do raise" and "don't lower", which share the same meaning of UP but are orthogonal to each other, were embedded in the lower area. Similarly, "do lower" and "don't raise" were embedded in the upper area. In other words, the X-OR problem was resolved in the PC4. Taken together, the model has extracted the X-OR problem that was implicitly included in the training data and learned to ground the orthogonal sentences in the same behavior thanks to its capability to embed input sequences non-linearly while maintaining

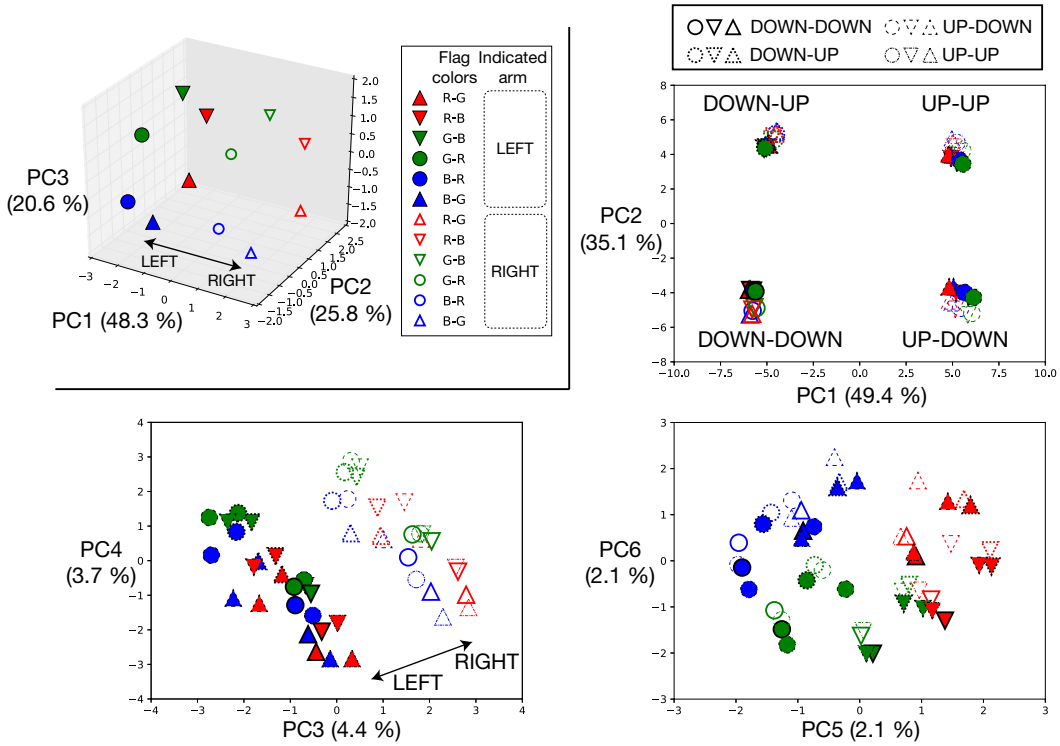

Figure 5.7. [Top left] The context states after receiving the sentence "do raise (word indicating the left flag color)" or "do raise (word indicating the right flag color)" projected onto the space spanned by PC1 to PC3. We fixed the robot's waiting posture to DOWN-DOWN, but the situations were different depending on the colors of held flags. For instance, the filled red triangle is the context state after giving the sentence "do raise red" to the model that was holding a red flag in the left arm and a green flag in the right arm (referred to as R-G). In the current task, which arm the robot must move was not determined by a given object word alone. Nevertheless, the PC1 represents which arm was referred to by the object word. This means that the model learned to encode an object word input and the visual context in an integrated manner and then achieved representations that correspond to a meaningful pair of "left" and "right". By decoding these representations, the model successfully chose an appropriate arm in every trial. [Others] The context states after receiving these sentences projected onto the space spanned by from PC1 to PC6. In these panels, all the situations not only about flag colors but also the waiting posture are included. Although point colors and shapes are as in the top-left panel, the frame lines vary in accordance with the arm posture. This visualization shows that the context state strongly reflected the current posture, so the PC1–2 space dominantly represented it. The representation of the "left–right" pair was still seen in the PC3–4 space. The visual context was also represented in the PC5–6 space, though the hexagonal shape was less clear.

the information that the given instructions were much different from each other in the other principal components.

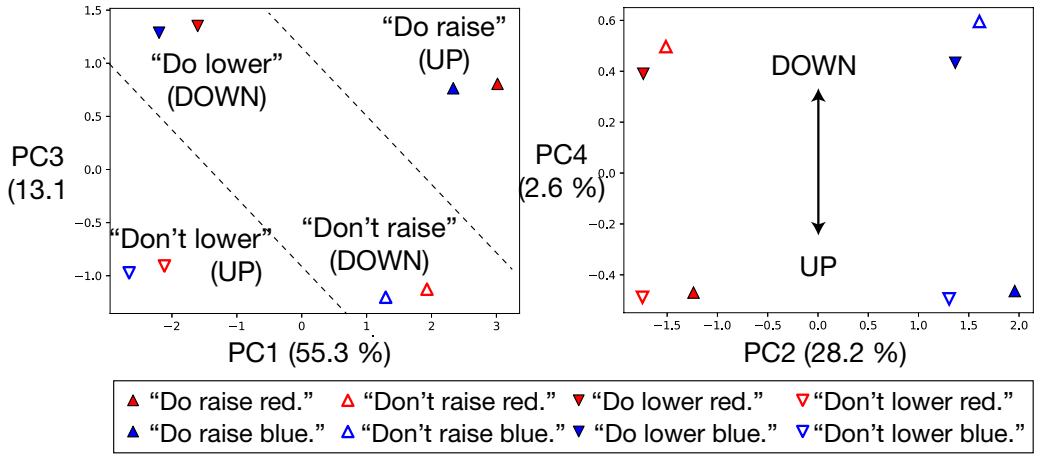

Figure 5.8. The context states after giving eight different single-object instructions to the model that was holding the red and blue flags in the left and right arms, respectively, and waiting in the DOWN-DOWN posture. The left panel shows the context states projected onto the PC1–3 subspace. The right one shows the context states projected onto the PC2–4 subspace. PC1, PC2, and PC3 directly correspond to the three parts of speech, i.e., the PC3, PC1, and PC2 axes represent the "do"/"don't", "raise"/"lower", and "red"/"blue" pairs, respectively. However, exploration of lower-rank components revealed that an X-OR problem caused by the "do"/"don't" and "raise"/"lower" pairs was resolved in the PC4 direction thanks to the model's capability to non-linearly transform the word inputs.

5.5.3 Representations of Logic Words: "And" and "Or"

Figure 5.9 illustrates the context states after giving the model that was holding the red and blue flags in the left and right arms, respectively, some different sentences whose form was "do raise (object part)." The object part was a single word, AND-concat, or OR-concat. AND-concat sentences that instructed the model to lift both arms up were embedded away from the representations of other sentences in the PC1 direction. The PC2 direction corresponds to the pair of "red" and "blue". Here, "or" that instructed the model to lift either arm was encoded in the middle area between representations of these two. This suggests that the representation of "or" was organized as an unstable area in the model's dynamical system and that as a result of these organized dynamics, behavior generation that apparently looked random every trial has been realized.

To verify this, we performed the following analysis. We gave the sentence "do raise green or blue" to the model with 2,048 different histories[5]. The top-

[5]More precisely, we gave this sentence to the model waiting in the DOWN-DOWN posture and holding green and blue flags in the left and right arms, respectively. However, in each of these

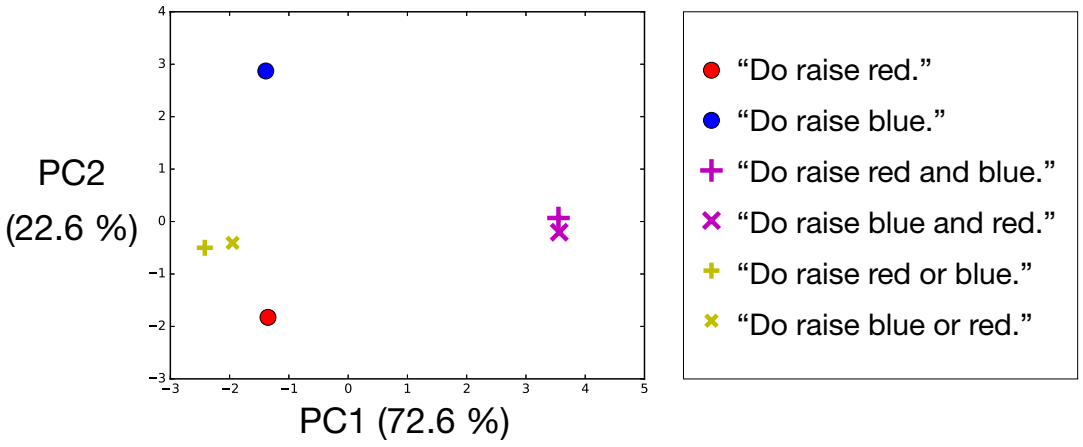

Figure 5.9. The context states after giving the model that was holding R-B flags some different sentences whose form was "do raise (object part)." The object part was a single word, AND-concat, or OR-concat. AND-concat sentences that instructed the model to lift both arms up were embedded away from the representations of other sentences in the PC1 direction. The PC2 direction corresponds to the pair of "red" and "blue". "Or," which instructed the model to lift either arm, was encoded in the middle area between representations of these two.

left panel of Figure 5.10 visualizes that the context states just after receiving the instruction were arranged in an arched area. Each point corresponds to a specific history. When the model encoded the instruction on the right side of the arch, it executed RIGHT-UP behavior. On the other hand, by encoding on the left side, the model executed LEFT-UP behavior. When the instruction was embedded in the middle area, unstable trajectory was produced. Nevertheless, in all trials, the context state finally converged to either a fixed-point attractor that represented the UP-DOWN or DOWN-UP posture, as shown in the bottom-right panel of Figure 5.10. Taken together, the model learned to respond to OR-concat instructions by representing them as the convergence from an unstable area to either one of multiple stable points.

2048 trials, the model had randomly different histories of the preceding episodes. Therefore, the context state also takes different values depending on different histories.

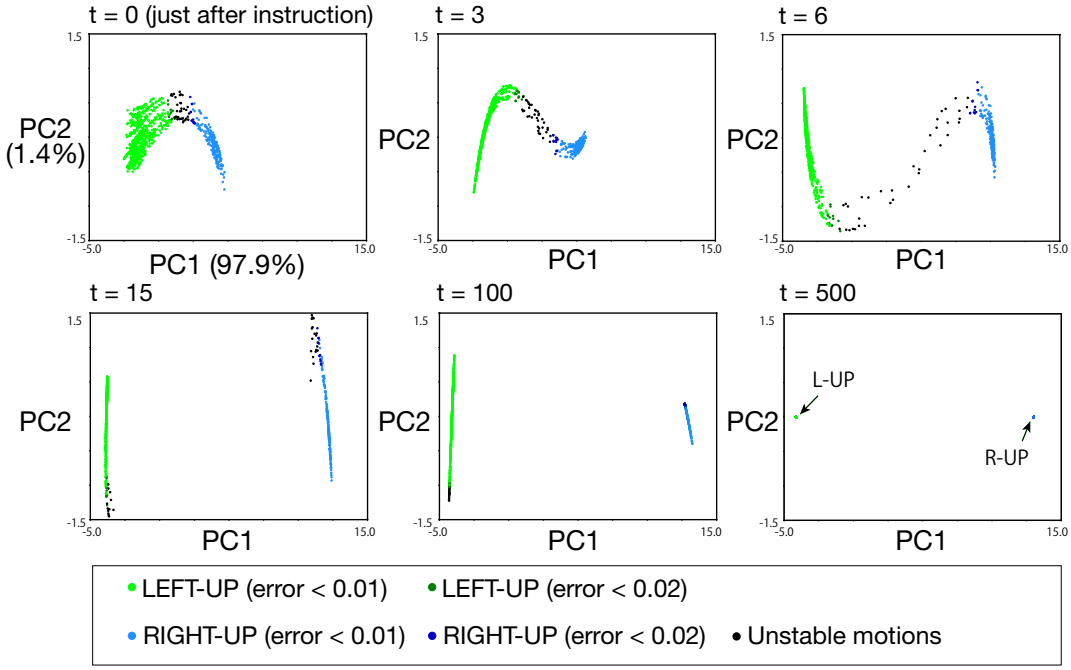

Figure 5.10. We gave the sentence "do raise green or blue" to the model that had 2,048 different histories of preceding episodes. The waiting posture and the flag colors were fixed in the DOWN-DOWN posture and G–B, respectively. The context states immediately after receiving the sentence were arranged in an arched area (top-left). Each point corresponds to a different history. When the model encoded the instruction on the right side of the arch, it executed RIGHT-UP behavior, and the context state converged in the fixed-point attractor that represented the DOWN-UP posture. On the other hand, by encoding on the left side, the model executed LEFT-UP behavior, and the context state converged in the fixed-point attractor representing the UP-DOWN posture. When the instruction was encoded in the middle area, an unstable trajectory was produced. Even in such cases, the context state finally converged to either fixed-point, as shown in the bottom-right panel.

5.5.4 Dynamical Representation of the Task Execution

Finally, we investigated how the model's internal dynamical system represented the task execution. Figure 5.11 visualizes the transition of the cell state while the model executed four successive episodes. The robot arms moved in the following order: DOWN–DOWN UP–DOWN, UP–UP, DOWN–UP, and DOWN–DOWN. The transitions from one posture to another one were internally represented as transitions between two different fixed-points (indicated by colored circles). Given a sentence, the cell state was activated and then reached a point marked by a cross

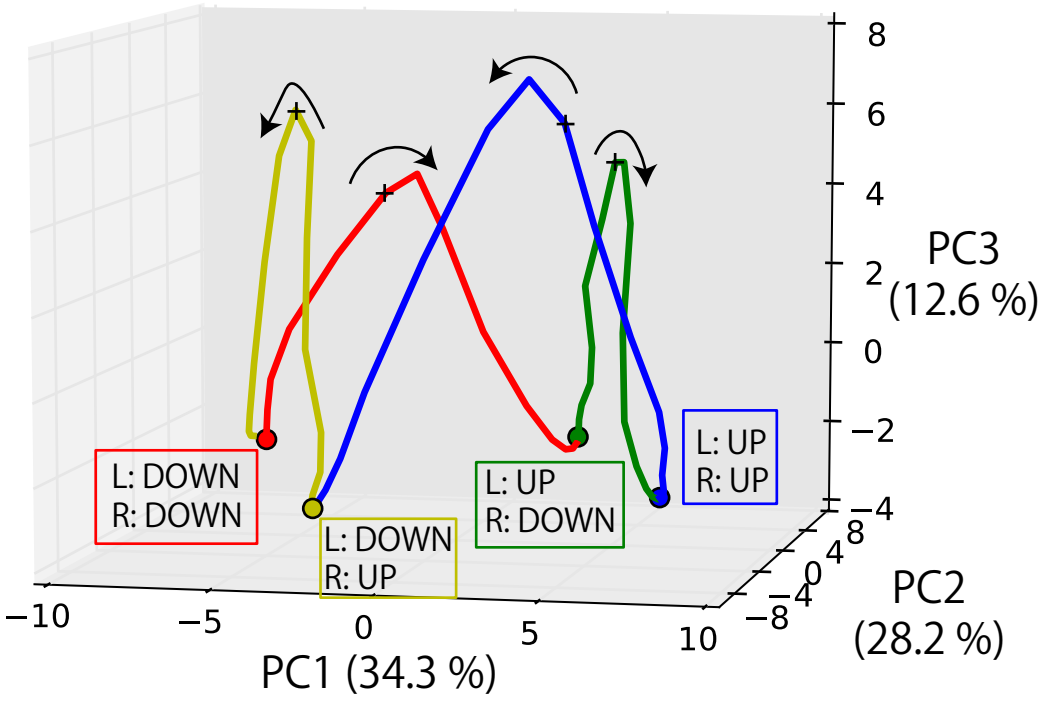

Figure 5.11. The context state transition while the model executed four successive episodes. The arm posture moved in the following order: DOWN-DOWN UP-DOWN, UP-UP, DOWN-UP, and DOWN-DOWN. Each transition from one posture to another one was internally represented as a transition between two different fixed-points (circle marks). Given a sentence, the cell state was activated and then reached a point indicated by a cross mark. By converging into another fixed-point from the activation, the model produced a behavior that would reach a posture corresponding to the fixed-point. After that, the model waited for a following instruction again.

mark[6]. From the marked state, by being attracted to one fixed-point again, the model produced a behavior that would reach a posture corresponding to the fixed-point. After that, the model waited for a following instruction. The same is true in cases in which the appropriate behavior was to maintain the current posture. While the produced joint trajectory was remaining stationary, the context state internally represented it as convergence into the original fixed-point after being activated in the PC3 direction.

[6]In fact, the model achieved these activations by integrating the current visual context with given content and logic words in the abovementioned manner, but it was difficult to visualize all of them together in one figure.

5.5.5 Comprehensive Summary of the Model Working

Taken together, the models learned to encode the instruction sentences into fixed-dimensional representations on the context layer in a form where the content words are grounded in the visual context and the robot's arm posture and where the logic words were encoded together with other content words in accordance with their own function as a logical operator. For instance, the words "do" and "don't", which indicate behaviors contrary to each other, transformed the model's cell state non-linearly to encode orthogonal sentences in the same area in a lower rank subspace. "Or," which requires the robot to execute behavior that appears random, was represented as an unstable area in the model's dynamics. Similarly to the experiment in the previous chapter, the task progress pattern, i.e., repetitions of instruction reception, behavior generation, and stand-by with multiple waiting postures, was also organized as the RNN's dynamical system together with grounding structure.

5.6 Further Analysis of Logic Words

In this chapter, we performed a learning experiment to evaluate whether the proposed framework can learn to convert instructions including both content words and logic words into behavioral sequences. We confirmed that the model successfully learned the given task, and we investigated how the model internally represented the content words and logic words together. However, in the task, when the instruction was in AND-concat or OR-concat form, the model did not need to consider the visual context. This is not natural in realistic situations. Therefore, in this section, we perform another learning experiment by designing a different task. In this task, the meanings of all the linguistic commands that include a logic word depend on the current visual context. We investigate whether the model organizes internal representations of logic words in a manner that reflects their own function even in such a realistic situation.

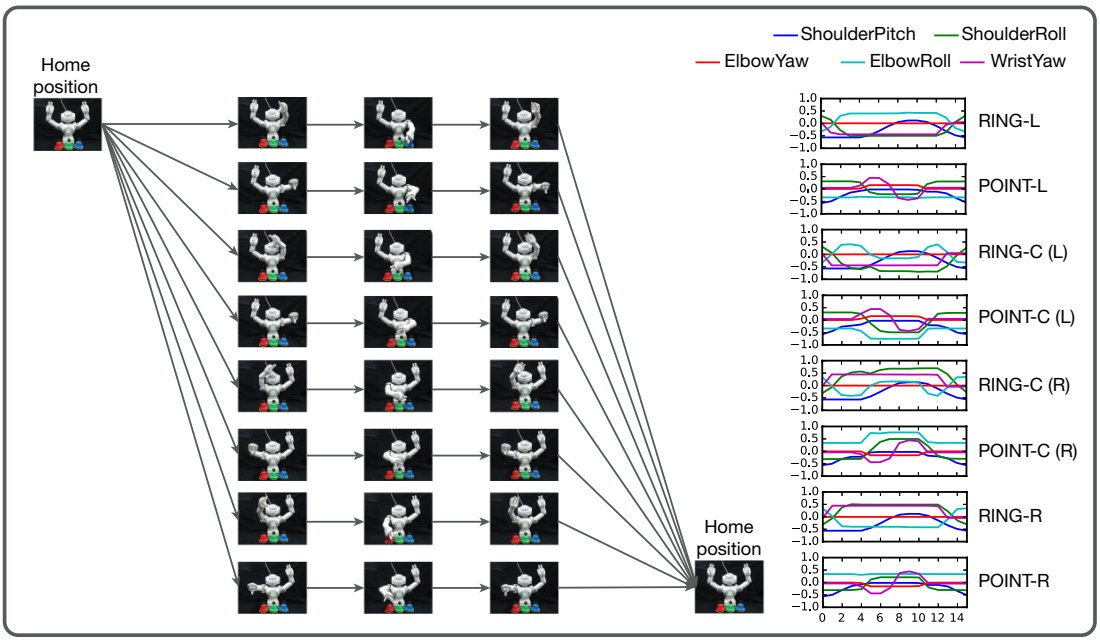

Figure 5.12. Illustration of behavioral sequences. All of these examples can be executed at SLOW and FAST speed.

5.6.1 Task Overview

This subsection describes the task design. First, we place three bells in front of the robot. They are differently colored as red, green, or blue and ordered randomly from the left to right. Next, we give an instruction that consists of a verb ("point", "ring"), an object ("red", "green", "blue"), and an adverb ("slowly", "fast") to the robot. When the object word indicates the left or right bell, the robot acts on it with the closer arm. When the object word indicates the center bell, the robot responds by acting on it with either arm. Figure 5.12 illustrates actual examples of arm motions.

Similarly to the flag up–down task, there are cases in which "and" concatenates two objects. In these situations, the robot responds by simultaneously ringing (pointing at) both bells. We also include OR-concat instructions in which the robot responds by ringing (pointing at) either one of the indicated bells. As another logic word, we sometimes prefix "not" to an object word. We refer to this type of instructions as NOT-prefixed. In these situations, the robot must respond by ringing (pointing at) two bells that are not indicated by the object.

For instance, when the robot receives the instruction "point not green fast," it must point at red and blue bells simultaneously at a fast speed.

5.6.2 Acquired Representations

The purpose of this additional experiment is to investigate how the model internally represents the logic words. Therefore, we report the details of the learning, such as model architecture and task performance, in Appendix B. This subsection directly describes the resulting representations after learning.

Representation of "Or"

We visualized the organized representations by PCA. First, the left panel of Figure 5.13 visualizes the internal states after receiving an instruction in the form of "ring (single-object) slowly" or "ring (OR-concat) slowly". We can see that the states after receiving an OR-concat instruction are located between the states after receiving a single-object instruction. For instance, "ring green or blue slowly" and "ring blue or green slowly" were encoded in the middle area between the embeddings of "ring green slowly" and "ring blue slowly." This suggests that the model represented "or" as unstable areas on the model's dynamical system, similar to in the flag up–down task. Actually, the right panel of Figure 5.13 visualizes that an arch-shaped encoding space was organized as in the flag up–down task, though the shape was less clean. It is notable that in contrast to the flag up–down task, where "or" always meant "left or right" regardless of the visual contexts, in the bell task, the meanings of OR-concat instructions depended on the given object words and the visual context. Even in such a grounded task, the functional meaning of "or" was successfully attained in a manner that was integrated with the object words and visual information.

Representations of "And" and "Not"

Figure 5.14 visualizes the internal states after receiving an instruction in the form of "ring (AND-concat) slowly" or "ring (NOT-prefixed) slowly". The bells were arranged in the fixed order of red, green, and blue from left to right. In the current

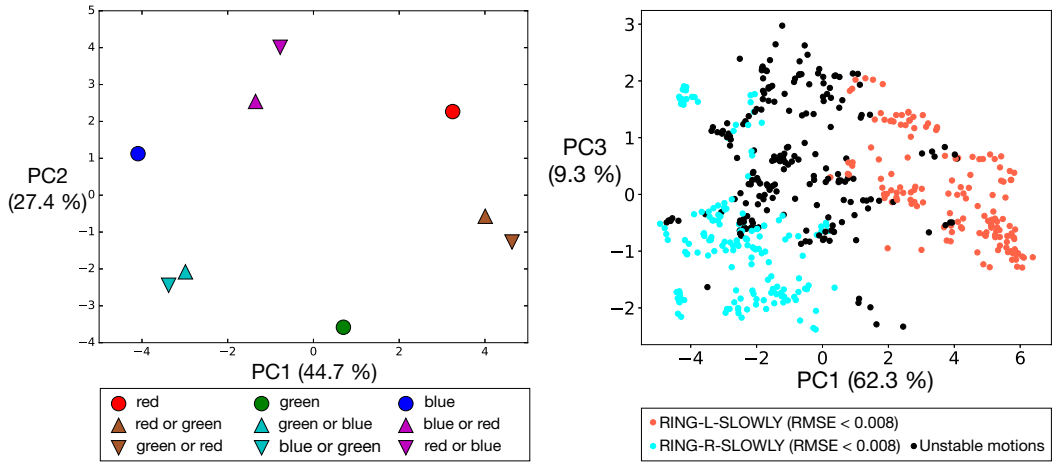

Figure 5.13. [Left] The cell states after receiving an instruction in the form of "ring (single-object) slowly" or "ring (OR-concat) slowly". The states after receiving an OR-concat instruction are located between the states after receiving a single-object instruction. For instance, "ring green or blue slowly" and "ring blue or green slowly" were encoded in the middle area between the embeddings of "ring green slowly" and "ring blue slowly". [Right] To the model that was waiting with bells placed in the order of red, green, and blue from left to right after 2,048 different contexts, we gave the sentence "ring red or blue slowly." The encodings of the sentence were arranged in an arch-shaped space. When the sentence was embedded on the left side of the arch, the model executed the RING-L-SLOW behavior. When encoding it on the right side, the model executed RING-R-SLOW behavior. When encoding in the middle area, the model produced an unstable behavior.

task, "not" refers to the complementary set of the stated color. For instance, "not blue" refers to the same bell set as "red and green." Although these two instructions have completely orthogonal object parts, they were embedded close to each other in the PC4–PC5 subspace. As a result, sentences that indicated the same behavior formed clusters, i.e., "red and green", "green and blue", and "blue and red" formed clusters. In contrast to the flag task, the instructions including "and" also required the model to take visual context into account. In such a grounded situation, the model successfully learned the function of "and" by encoding it in a way integrated with the object word and visual information.

Taken together, even in the task that simultaneously required both grounding in the current context and processing of logic words, the functional meanings of logic words were successfully represented in a manner that was integrated with other content words.

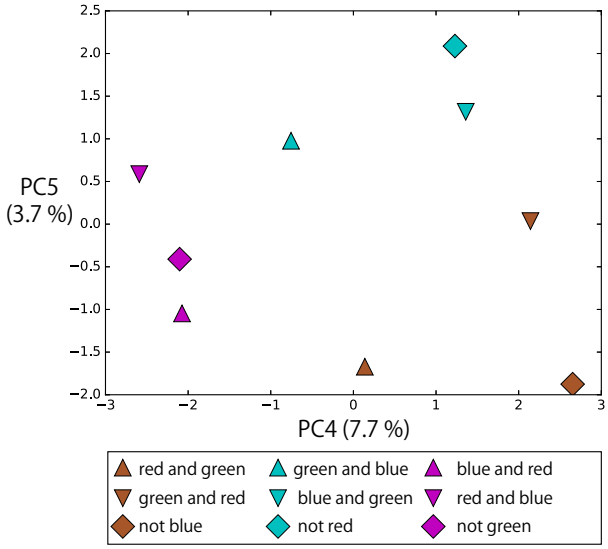

Figure 5.14. The cell states after receiving an instruction in the form of "ring (AND-concat) slowly" or "ring (NOT-prefixed) slowly". The bells were arranged in the fixed order of red, green, and blue from left to right. In the current task, "not" refers to the complementary set of the stated color. For instance, "not blue" refers to the same bell set as "red and green." Although these instructions have completely orthogonal object parts, they were embedded close to each other in the PC4–PC5 subspace. As a result, sentences that indicated the same behavior formed clusters, i.e., "red and green", "green and blue", and "blue and red" formed clusters.

5.7 Discussion

This chapter demonstrated that the proposed EDM is able to learn to convert linguistic instructions consisting of both content words and logic words into behavioral sequences. We also investigated what kind of compositional representations of content words and logic words emerged from the experience of the interactive task. In the case of content words, an object word was encoded by being integrated with visual context. A verb was grounded in different joint trajectories depending on the robot's current arm state. Simultaneously, the model also successfully dealt with the logic words "do", "don't", "and", and "or". By encoding logic words together with other content words in a manner that reflected their functions as logical operators, the model successfully executed appropriate behaviors. This section discusses three types of logic words employed in the current task.

5.7.1 NOT: as Nonlinear Transformation of Internal States

In the flag up–down task, the model understood "raise" and "lower" as the goal-oriented behaviors UP and DOWN, respectively, by combining them with "do" and "don't" in an X-OR manner. The model encoded these orthogonal phrases as sharing the same meaning in a relatively low rank component thanks to its ability to non-linearly transform input. In the same way, "not" in the bell task worked as an operation to choose a complementary set. In the field of NLP, Li et al. [102] performed a similar kind of analysis on a deep NN model. They demonstrated that a deep NN optimized for sentiment analysis drastically changes its latent representation by receiving a negation word. For instance, "not good" was embedded closer to "bad" than to "good." However, our analysis demonstrated that even though the combined meaningful representation corresponding to the goal-oriented behavior is achieved in a low rank component, the information that input sentences were perfectly different was still maintained in most principal components. In other words, not only were meaningful representations gained by integrating element words and sensorimotor information but also information of element words was still stored in the memory cells.

This aspect seems to be important. For instance, imagine a case of the bell task in Chapter 4 where the model embeds both the instructions "point green slowly" for a B–G arrangement and "point red slowly" for a G–R arrangement as composed representations corresponding to the POINT-RIGHT-SLOW behavior by losing the information about element words. In this case, the robot could not adaptively behave in response to changes, e.g., a sudden relocation of bells during behavior execution, because the model has already lost information about the given object word. By remembering given element words, adaptive response to such fluctuations would be achieved, although it is not certain that the model trained in the current task is able to deal with such situations because training data did not include such situations.

Recently, EDM models with attention mechanisms that store the representations acquired by the encoder on a memory and refer them during the decoding phase have been used more often than the normal type of sequence to sequence model and have achieved better results ([26, 27]). In comparison with question

answering or translation tasks with no noise, an attention mechanism that can explicitly maintain information about compositional elements seems to be essentially required in robot learning tasks in which the model is always exposed to noise and fluctuations during behavior generation. In future work, we should therefore evaluate the effectiveness of attention mechanisms in robot tasks that require language understanding.

5.7.2 AND: as a Universal Quantifier

In the flag up–down task, "and" itself worked as a sort of universal quantifier without grounding in color words. When the model holding the blue and red flags received "do raise blue and green," it raised both arms. The results were similar in other contradictory situations. In other words, when the instruction is AND-concat, the model ignored object words. In these cases, only a verb (and a truth value) is grounded in the behavior to be generated. In that sense, the model represented "and" as a concept one step higher. Before the experiment, we did not expect such interpretation of "and" by the model. This is not our common usage of "and", however it seems to be a rational and reasonable solution in the frame of the flag up–down task.

However, in practical situations, we use "and" in different manners to combine not only multiple words but also phrases and sentences. Future works must investigate how RNNs can handle and represent such higher order or general types of "and." Most recently, a series of studies has reported that the straightforward end-to-end learning of RNNs with the gradient descent method did not give them hierarchically generalizable understanding of function words or syntactic compositionality [103, 104, 105] although they trained RNNs purely with sequences of symbols. In fact, we might have to improve the model architecture or learning algorithm to achieve such systematic level generalization. However, we also believe that to achieve generalizable understanding of function words, embodied experiences are essentially required beyond purely symbolic experiences. Forster et al. [106] demonstrated that small humanoid robot called iCub acquired the utilization of a negative word such as "no" from interaction with a human caregiver. This was done by the caregiver uttering negative sentences while observing iCub's

rejection of objects or while prohibiting iCub from reaching objects. Although this experiment is an example specialized to negation, we think that such embodied experiences help robots to achieve a generalizable understanding of function words.

5.7.3 OR: as Unstable States in RNN's Dynamical System

In the flag up–down task, the model trained with noiseless joint angle data resulted in worse performance than with noise. In a preliminary experiment in which OR-concat sentences were not included, this difference was not observed. This suggests that the inclusion of ambiguous OR-concat sentences, which gave either of two answers as correct randomly every episode, made the learning by the back-propagation algorithm unstable. This reminds us of the famous thought experiment of Buridan's donkey. In this experiment, a human places a stack of hay on a donkey's left and right sides at precisely the same distance away. Confronting the dilemma, the donkey was not able to choose which side to go and eventually died of hunger. The current analysis suggests that the model successfully resolved this dilemma, which the donkey confronted too honestly, by making use of the small noise on target joint angles as a clue to decide which arm to move and by self-organizing an unstable area in the dynamical system, which converges into either of two fixed-points.

In future work, we must perform a more detailed investigation of the characteristics of the model's dynamical system. In existing studies, Tani and Fukumura [66] demonstrated that a deterministic RNN has the capability of regenerating symbol sequences that follow simple stochastic rules by self-organizing a chaotic dynamical system. In Namikawa et al. [67], an MTRNN had the capability of replicating pseudo-stochastic transitions among multiple robot motion primitives. The current experiment demonstrated that similar functions to execute behaviors as though they were executed in a probabilistic manner can be acquired from the experience of a language grounding task. The current result also demonstrated that the increase of LSTM nodes improves the capability to respond to OR-concat instructions. We think that the increase of the number of nodes improves the model's representation ability and leads the model into learning to

forcibly embed the stochastic experience in a chaotic dynamical system.

5.7.4 Encoding Both Content Words and Function Words as Fixed-dimensional Representations

The experiment in this chapter included not only content words but also logic words as examples of function words. Content words were understood in an integrated manner together with visual and proprioceptive information. On the other hand, the logic words were encoded as operations on the representations of content words. In this way, a meaning of a sentence was encoded together with the current context. RNNs that learn to encode all words and multimodal information into a fixed-dimensional distributed representation in a unified manner have an advantage over probabilistic modeling methods. Although probabilistic models [55, 57, 40] are also able to learn the stochastic co-occurrence relationships between content words and multimodal features, it is difficult for them to learn to model how logic words combine the grounded meanings of other content words. The most typical examples are negative expressions, such as "no" and "not". The functions of these words cannot be captured as simple co-occurrence relationships. In contrast, RNNs can learn to model their function as a non-linear transformation of their context state.

In this chapter, we focused only on some logic words as examples of function words. However, our language systems include many other kinds of function words, such as prepositions, pronouns, and articles. We must investigate how our proposed model can learn these other types of function words as well in future work.

5.8 Summary

This chapter performed experiments to evaluate whether the proposed system can learn to convert linguistic instructions, including both content words and function words, into behavioral sequences. The learning results showed that the proposed method is actually able to ground sentences that consist of both types

of words in a behavioral sequence with a certain level of generalization. We also investigated what kind of compositional structures for such grounding emerged from the experiences of an interactive task. Content words were encoded in a way grounded in the visual context and the robot's current posture. The logic words were represented together with other content words in accordance with their own functions as logical operators. The words "do" and "don't", which indicate behaviors contrary to each other, worked as non-linear transformations of context states to encode orthogonal sentences into the same area in a low-rank component. In the flag up–down task, "and" eliminated grounding in the vision in a reasonable manner and worked as if it was a universal quantifier. The function of "or," which instructed the model to generate behavior that looked apparently random, was represented as unstable areas of the model's internal dynamical system. Figure 5.15 summarizes the main targets discussed in this chapter.

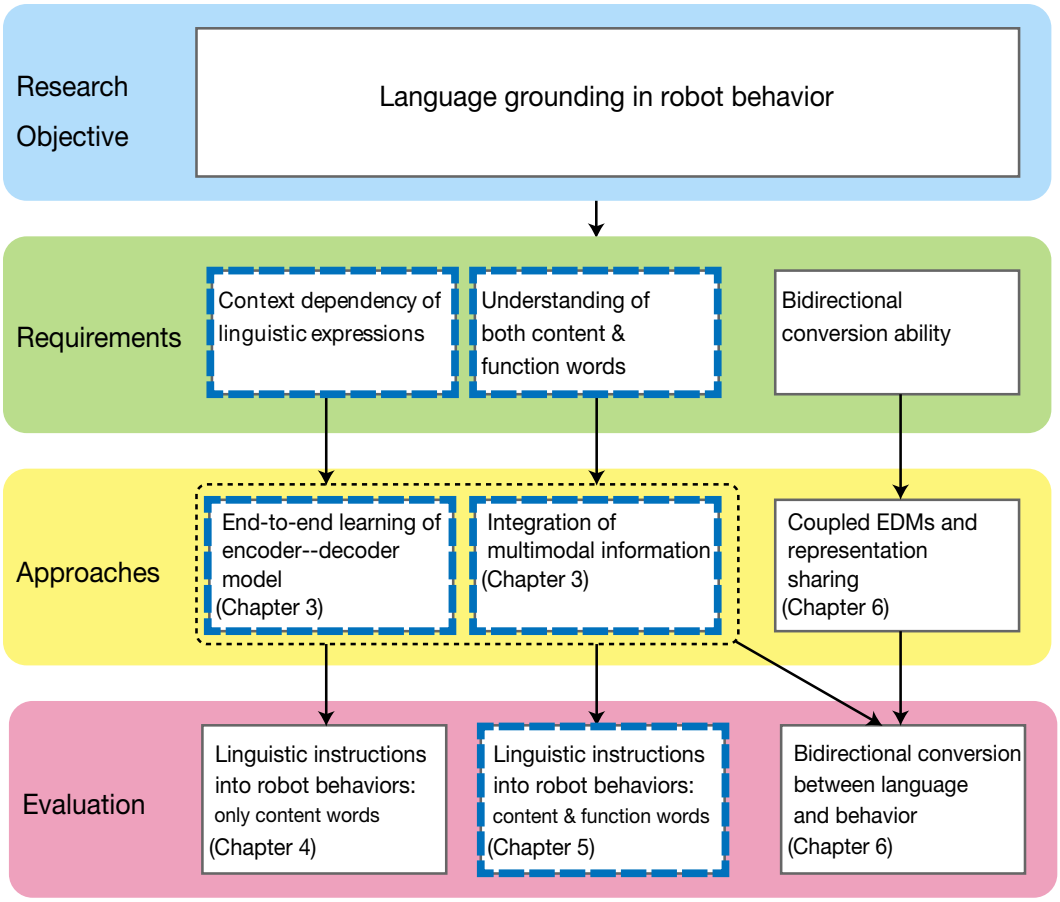

Figure 5.15. The main targets discussed in Chapter 5.

Chapter 6

Bidirectional Conversion between Language and Behavior

6.1 Introduction

In the previous two chapters, through simple robot experiments, we confirmed that EDMs extended with multimodal input can **convert linguistic instructions into robot behavioral sequences appropriate for the current context (R1)**. Specifically, the experiment in Chapter 5 demonstrated that **sentences consisting of both content words and logic words (R2)** could be successfully grounded in behavior. This chapter proposes to extend the framework to address the remaining issue we have not touched on yet, namely **bidirectional conversion capability (R3)**.

Most conventional studies on learning between language and robot behavior have dealt predominantly with only unidirectional conversion. Most recently, deep learning methods have been used for language grounding tasks in a simulated environment [15, 16] and have achieved outstanding performances. However, in general, NNs (including ones with deep architecture) are optimized for a unidirectional task from input to output, i.e., from language instructions to action sequences. In this chapter, to achieve bidirectional mapping between language and robot behavior, we extend our proposed EDM to a two-coupled form and train these two EDMs so that their representation spaces are shared. In this extended framework, language and robot behavioral sequences can be bidirectionally

converted through the shared latent space.

The remainder of this chapter is organized as follows. In Section 6.2, we propose a learning framework to bidirectionally map between linguistic expressions and robot behavior. In Section 6.3, we describe the setup of an evaluation robot experiment, namely task details and target data configuration. Section 6.4 shows task performance results after learning. In Section 6.5, we analyze the model's internal representations that link language and robot behavior. Section 6.6 compares the proposed method with existing bidirectional methods and considers future directions for the model improvement. Section 6.7 concludes this chapter.

6.2 Bidirectional Conversion by Coupled EDMs and Representation Sharing

6.2.1 Overview of the System

This section proposes an NN architecture and learning algorithm that allows a robot to generate both (i) behavioral sequences responding to multi-word instructions and (ii) multi-word descriptions given its own behavioral sequence. Here, we assume that we have a paired dataset of language and behavior. Therefore, the method proposed here is based on the S-EDM.

As shown in Figure 6.1, the model is comprised of coupled recurrent autoencoders (RAEs) [107]: one is for language, and the other is for behavior. The RAE is an EDM, so it consists of an encoder network and a decoder network. In the learning process, RAEs learn to minimize the error between a generated sequence and the original sequence (referred to as reconstruction loss). In other words, they learn to embed sequential data into fixed-dimensional latent representations ($\boldsymbol{z}^{\mathrm{lng}}$ or $\boldsymbol{z}^{\mathrm{bhv}}$) in a manner that enables the embedded sequences to be regenerated from the representations by utilizing their decoder.

If we train two EDMs independently, the organized space that efficiently embeds behavioral sequences may differ from the space for their descriptions, as described as "arbitrariness" of language by Saussure [97]. Thus, to make these spaces organize the shared structure, we introduce another loss term, which binds

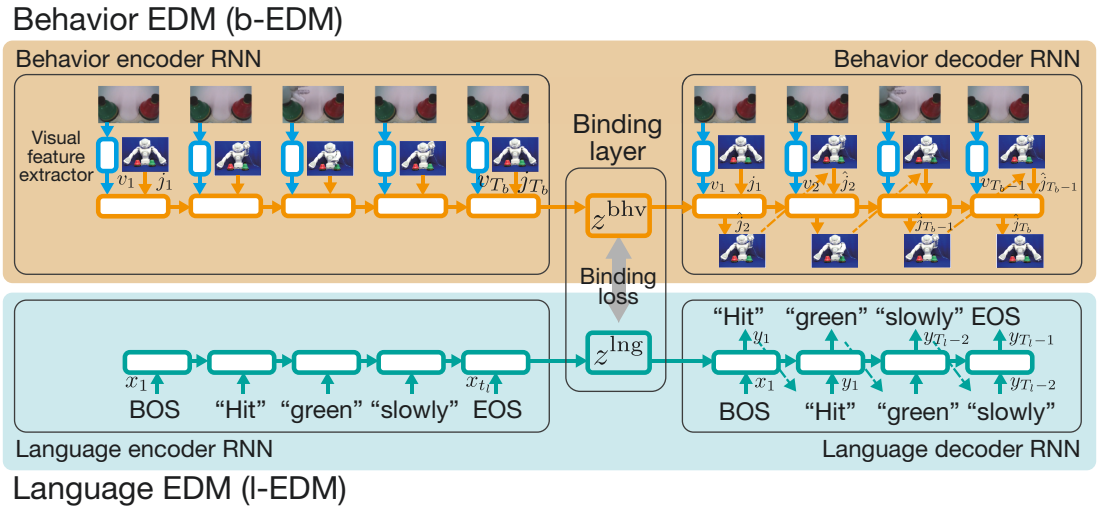

Figure 6.1. Overview of the coupled EDM framework to convert language and behavior bidirectionally.

representations of a behavioral sequence and its paired sentence to be mutually close in the latent space. By minimizing both reconstruction loss and representation distance together, we expect the model to achieve the capability of bidirectionally converting between language and behavior. Generating behavioral sequences responding to instructions is performed by utilizing the encoder of the language EDM (l-EDM) to embed an instruction and utilizing the decoder of the behavior EDM (b-EDM) to decode the representation. On the other hand, the model generates a linguistic description for a behavioral sequence by utilizing the encoder of the b-EDM to embed a behavioral sequence and then utilizing the decoder of the l-EDM to decode the representation.

6.2.2 EDM for Language

To encode word sequences, we use an l-EDM (the lower part of Figure 6.1). The l-EDM is comprised of an encoder and a decoder; both of which are RNNs. The encoder network encodes a word sequence of length T_l, $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{T_\mathrm{l}})$ as a fixed-dimensional representation $\boldsymbol{z}^{\mathrm{lng}}$ as follows:

$$\boldsymbol{h}_t^{\mathrm{enc}} = \mathrm{EncRNNCell}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}^{\mathrm{enc}}) \quad (1 \leq t \leq T_\mathrm{l}), \tag{6.1}$$

$$\boldsymbol{z}^{\mathrm{lng}} = \boldsymbol{W}^{\mathrm{enc}} \boldsymbol{h}_{T_\mathrm{l}} + \boldsymbol{b}^{\mathrm{enc}}. \tag{6.2}$$

Here, the EncRNNCell function denotes a learnable recurrent cell and $\boldsymbol{h}_t^{\mathrm{enc}}$ is the output of EncRNNCell at time step t. The final cell output $\boldsymbol{h}_{T_\mathrm{l}}$ is projected into the shared space by a learnable weight $\boldsymbol{W}^{\mathrm{enc}}$ and a learnable bias $\boldsymbol{b}^{\mathrm{enc}}$. We initialize $\boldsymbol{h}_0$ with a zero-filled vector.

After that, the decoder network produces a sequence by recursively decoding the latent vector $\boldsymbol{z}^{\mathrm{lng}}$ as follows:

$$\boldsymbol{h}_0^{\mathrm{dec}} = \boldsymbol{W}^{\mathrm{dec}}\boldsymbol{z}^{\mathrm{lng}} + \boldsymbol{b}^{\mathrm{dec}}, \tag{6.3}$$

$$\boldsymbol{h}_t^{\mathrm{dec}} = \mathrm{DecRNNCell}(\boldsymbol{y}_{t-1}, \boldsymbol{h}_{t-1}^{\mathrm{dec}}) \quad (1 \leq t \leq T_\mathrm{l} - 1), \tag{6.4}$$

$$\boldsymbol{y}_t = f(\boldsymbol{W}^{\mathrm{out}}\boldsymbol{h}_t^{\mathrm{dec}} + \boldsymbol{b}^{\mathrm{out}}) \quad (1 \leq t \leq T_\mathrm{l} - 1). \tag{6.5}$$

Here, DecRNNCell denotes a learnable recurrent cell, and $\boldsymbol{h}_t^{\mathrm{dec}}$ is the output of DecCell at time step t. $\boldsymbol{W}^{\mathrm{dec}}$ and $\boldsymbol{b}^{\mathrm{dec}}$ denote a learnable weight and a bias, respectively, to transform the latent vector $\boldsymbol{z}^{\mathrm{lng}}$ into the initial context of the decoder $\boldsymbol{h}_0^{\mathrm{dec}}$. Similarly, $\boldsymbol{W}^{\mathrm{out}}$ and $\boldsymbol{b}^{\mathrm{out}}$ are a learnable weight and a bias, respectively, to transform the output of DecRNNCell into the vocabulary size vector. f denotes an activation function. This study represents words in the one-hot form. Thus, the softmax function is used as f. The decoder receives a vector that represents the beginning of sequence (BOS) signal in place of $\boldsymbol{y}_0$.

The learning target for the l-EDM is to reconstruct an original word sequence given to the encoder from the latent representation. Therefore, the loss function to be minimized is defined as the cross entropy between the input and the output:

$$L_{\mathrm{lng}} = \frac{1}{T_\mathrm{l} - 1}\sum_{t=1}^{T_\mathrm{l}-1}\Big(-\sum_{w}^{W} x_{t+1}(w)\log y_t(w)\Big), \tag{6.6}$$

where W denotes the vocabulary size. Similarly to the case of EDMs proposed in Chapter 3, which unidirectionally convert language to behavior, all the learnable parameters are optimized by gradient descent algorithm. This unsupervised framework trains the model to organize a latent feature space in which a given sentence dataset is efficiently embedded.

6.2.3 EDM for Behavior

To encode behavioral sequences, we employ a b-EDM (the upper part of Figure 6.1). Similarly to the l-EDM, the b-EDM is comprised of an encoder module and a decoder module. A behavioral sequence is represented as a series of length T_b, $(\boldsymbol{j}_1, \boldsymbol{j}_2, ..., \boldsymbol{j}_{T_\mathrm{b}})$ of joint angles and a vision sequence $(\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_{T_\mathrm{b}})$ that accompanies it. Here, we apply a visual feature extractor that compresses raw images into low-dimensional feature vectors before feeding them into the b-EDM (Figure 6.1). The appropriate extractor would depend on the tasks. For instance, we employ a convolutional NN in Section 6.3.

The encoder of the b-EDM embeds a sequence $((\boldsymbol{j}_1; \boldsymbol{v}_1), (\boldsymbol{j}_2; \boldsymbol{v}_2), ..., (\boldsymbol{j}_{T_\mathrm{b}}; \boldsymbol{v}_{T_\mathrm{b}}))$, which is a concatenation of joint angle vector and visual feature vector, as a latent representation $\boldsymbol{z}^\mathrm{bhv}$. The model structure and the forward propagation of the encoder are similar:

$$\boldsymbol{h}_t^\mathrm{enc} = \mathrm{EncRNNCell}(\boldsymbol{v}_t, \boldsymbol{j}_t, \boldsymbol{h}_{t-1}^\mathrm{enc}) \quad (1 \leq t \leq T_\mathrm{b}), \tag{6.7}$$

$$\boldsymbol{z}^\mathrm{bhv} = \boldsymbol{W}^\mathrm{enc} \boldsymbol{h}_{T_\mathrm{b}}^\mathrm{enc} + \boldsymbol{b}^\mathrm{enc}. \tag{6.8}$$

On the other hand, the input and output of the decoder are different:

$$\boldsymbol{h}_0^\mathrm{dec} = \boldsymbol{W}^\mathrm{dec} \boldsymbol{z}^\mathrm{bhv} + \boldsymbol{b}^\mathrm{dec}, \tag{6.9}$$

$$\boldsymbol{h}_t^\mathrm{dec} = \mathrm{DecRNNCell}(\boldsymbol{v}_t, \widehat{\boldsymbol{j}}_t, \boldsymbol{h}_{t-1}^\mathrm{dec}) \quad (1 \leq t \leq T_\mathrm{b} - 1), \tag{6.10}$$

$$\widehat{\boldsymbol{j}}_{t+1} = f(\boldsymbol{W}^\mathrm{out} \boldsymbol{h}_t^\mathrm{dec} + \boldsymbol{b}^\mathrm{out}) \quad (1 \leq t \leq T_\mathrm{b} - 1). \tag{6.11}$$

Equations (6.10) and (6.11) express that the decoder predicts only joint angle values at the next time step $\boldsymbol{j}_{t+1}$ by producing $\widehat{\boldsymbol{j}}_{t+1}$. At every time step, the model receives the visual features as external input (e.g., teacher forcing). In contrast, the joint input nodes receive joint angle values predicted by the decoder itself at the previous time step (i.e., free running)[1]. We defined the loss term to train the b-EDM as the MSE between the predicted joint angles and the ground truth:

$$L_\mathrm{bhv} = \frac{1}{T_\mathrm{b} - 1} \sum_{t=1}^{T_\mathrm{b}-1} ||\boldsymbol{j}_{t+1} - \widehat{\boldsymbol{j}}_{t+1}||_2^2. \tag{6.12}$$

[1]At the initial step, the initial robot posture $\boldsymbol{j}_1$ was set instead of the previous prediction $\widehat{\boldsymbol{j}}_1$.

The reason we built the b-EDM to generate joint angles only is that we want the model to learn to solve context dependency of language grounding in behavior. Similarly to the unidirectional model proposed in Chapter 3, the context dependency between language and behavior will be solved by receiving visual context externally. The following describes the models mechanism in the form of a concrete example. Actual joint trajectories that realize the instruction "pick the green box" vary from each of the other tokens depending on the position of the green box. These various behavioral trajectories are compressed by the encoder of the b-EDM and bound with the unique representation of the sentence "pick the green box" compressed by the encoder of the l-EDM. Consequently, these diverse behavioral trajectories are not encoded as ones that differ from each other but are embedded close to each other as sequences that are semantically the same, namely "pick the green box". Thus, the b-EDM decoder generates an appropriate trajectory by integrating such a semantic encoding and the current visual context.

6.2.4 Representation Sharing

To bind the representations of behavioral sequences and their corresponding sentences, we introduce another loss term L_{shr} in addition to the reconstruction losses L_{lng} and L_{bhv}. We denote a batch of latent representations of behavioral sequences as $\{\boldsymbol{z}_i^{\mathrm{bhv}} | 1 \leq i \leq N\}$ and the representations of their paired sentences as $\{\boldsymbol{z}_i^{\mathrm{lng}} | 1 \leq i \leq N\}$. Here, N denotes the batch size. We introduce the following loss to bind representations:

$$
\begin{aligned}
L_{\mathrm{shr}} = & \sum_{i}^{N} \psi(\boldsymbol{z}_i^{\mathrm{bhv}}, \boldsymbol{z}_i^{\mathrm{lng}}) + \\
& \sum_{i}^{N} \sum_{j \neq i} max\{0, \Delta + \psi(\boldsymbol{z}_i^{\mathrm{bhv}}, \boldsymbol{z}_i^{\mathrm{lng}}) - \psi(\boldsymbol{z}_i^{\mathrm{bhv}}, \boldsymbol{z}_j^{\mathrm{lng}})\}, \qquad (13)
\end{aligned}
$$

where ψ computes a distance (or dissimilarity) between vectors. The first term forces representations of a behavioral sequence and its corresponding sentence to be close to each other. On the other hand, the second term forces the representation of a behavior (resp., sentence) to be far from that of its unpaired sentences

(resp., behaviors) when the distance between them is smaller than that from the paired sentence (resp., behavior). Δ is a scalar value that denotes a margin added to the distance between paired representations to enhance the loss.

6.2.5 Learning Procedure

We optimize the trainable parameters by stochastic gradient descent algorithm. Algorithm 1 describes the whole learning process. α, β, and γ are hyperparameters that control the importance of each loss function. A denotes the learning rate, which is constant or adaptive to the learning progress (e.g., Adam [82]).

Algorithm 1 Learning of coupled RAEs

Require: $X^{\mathrm{lng}}, X^{\mathrm{bhv}}$: paired dataset
Require: α, β, γ, A: hyperparameters
  initialize trainable parameters randomly: θ
  while not done **do**
    Sample a batch $\{x_i^{\mathrm{lng}}, x_i^{\mathrm{bhv}}\}$ randomly from $X^{\mathrm{lng}}, X^{\mathrm{bhv}}$
    Calculate $L_{\mathrm{lng}}, L_{\mathrm{bhv}}, L_{\mathrm{shr}}$ by forward path
    Compute total loss: $L_{\mathrm{all}} \Leftarrow \alpha L_{\mathrm{lng}} + \beta L_{\mathrm{bhv}} + \gamma L_{\mathrm{shr}}$
    Compute gradients $\nabla_\theta L_{\mathrm{all}}$ by backward path
    Apply the gradients $\theta \Leftarrow \theta - A \nabla_\theta L_{\mathrm{all}}$
  end while

6.3 Experimental Setup

6.3.1 Task Design

We carried out a learning experiment using a real robot to evaluate the bidirectional conversion performance of our proposed framework. First, we put two out of three boxes — each of which colored red, green, or yellow — in front of the robot in fixed positions (Figure 6.2). Thus, ${}_3P_2 = 6$ box arrangement patterns are possible. In each situation, the robot could execute the 12 behaviors listed in Table 6.1. We determined a sentence corresponding to each behavior in accordance with the box arrangement. More precisely, each sentence is comprised of a verb ("push", "pull", "slide"), an object ("red", "green", "yellow"), and an

Table 6.1. List of behaviors

Behavior name	Details
PUSH-L-SLOW	Push a left box toward the front slowly
PUSH-L-FAST	Push a left box toward the front fast
PUSH-R-SLOW	Push a right box toward the front slowly
PUSH-R-FAST	Push a right box toward the front fast
PULL-L-SLOW	Pull a left box slowly
PULL-L-FAST	Pull a left box fast
PULL-R-SLOW	Pull a right box slowly
PULL-R-FAST	Pull a right box fast
SLIDE-L-SLOW	Slide a left box to the right slowly
SLIDE-L-FAST	Slide a left box to the right fast
SLIDE-R-SLOW	Slide a right box to the left slowly
SLIDE-R-FAST	Slide a right box to the left fast

adverb ("slowly", "fast"). Thus, $3 \times 3 \times 2 = 18$ sentences are possible. A verb and an adverb are assigned only depending on behavior type, regardless of the box colors. On the other hand, an object word is determined depending on the box colors. We assign the color subject to the behavior to the object word. For instance, when a yellow box and a red box are put on the left and right, respectively, the sentence describing PULL-L-FAST behavior is "pull yellow fast" and that describing PUSH-R-SLOW behavior is "push red slowly". Due to such task design, the number of possible behavior types and that of possible descriptions are different.

6.3.2 Target Data

In advance of data recording, we preprogrammed the 12 behavioral trajectories by computer. For each of the six box combinations, we made the robot execute these 12 behaviors while recording 10-DOF joint angle values on both arms together with images (width: 160, height: 120, channels: RGB) from a built-in camera every 300 ms. SLOW and FAST behaviors were recorded as a sequence of approximately 39 and 26 frames, respectively. We recorded the possible 72 patterns (12 behaviors with six box combinations each) six times.

After data collection, we built a convolutional autoencoder (CAE) [108], trained

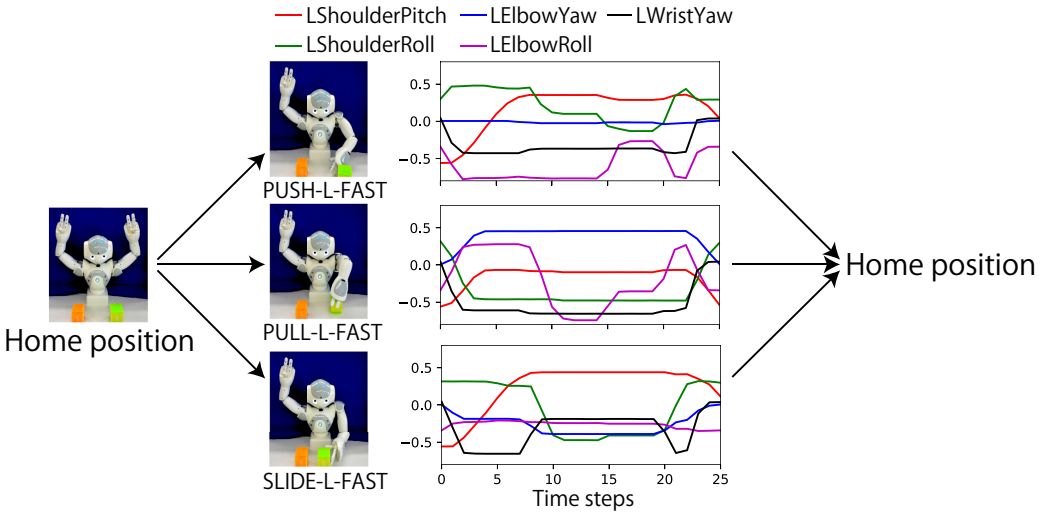

Figure 6.2. Examples of behavioral sequences (only left-arm joints are plotted).

Table 6.2. Model configuration of the CAE employed.

Layer type	Layers
Input (height, width, num_channels)	(120, 160, 3)
Convolution2D (num_channels, kernel size, stride)	- (8, 4, 2) - (16, 4, 2) - (32, 4, 2) - (64, 8, 5)
Full connection (n_units)	- (384) - (192) - (10) - (192) - (384)
Deconvolution2D (num_channels, kernel size, stride)	- (32, 8, 5) - (16, 4, 2) - (8, 4, 2) - output (3, 4, 2)

it from scratch using the collected images, and employed it as a visual feature extractor. The CAE compressed the images into 10-dimensional feature vectors on its center layer. We used them as visual inputs for the b-RAE. Table 6.2 reports the details of the employed CAE architecture.

The descriptions were formed as a sequence of one-hot vectors. Every description includes <BOS> and <EOS> signals at the beginning and the end, respectively.

Table 6.3. Hyperparameters of the model and learning

Hyperparameter	Value
l-RAE encoder	single-layer LSTM (100 nodes)
l-RAE decoder	single-layer LSTM (100 nodes)
b-RAE encoder	double-layer LSTM (100 nodes per layer)
b-RAE decoder	double-layer LSTM (100 nodes per layer)
Shared nodes	100 nodes
Distance function ψ	Euclidean distance
Margin Δ	1.0
Loss mixing rates α, β, γ	all 1.0
Optimizer	Adam (learning rate: 0.001)
Batch size	50
, Learning iteration	20,000

6.4 Learning Results

6.4.1 Training Details

We separated the 72 possible episode patterns into two parts: 54 for training and 18 for testing[2]. The hyperparameters of the model and learning are listed in Table 6.3.

6.4.2 Task Performance

Conversion from Behavior to Language

First, we evaluated the model's capability to generate descriptions of given behavioral sequences. As stated in Section 6.2, this conversion is carried out by utilizing the b-EDM to embed a behavioral sequence and then utilizing the decoder of the l-EDM to decode the representation. The l-EDM predicts the probability distribution over the vocabulary every time step. At each time step, we accepted a word corresponding to the node that has taken the maximum value as the model's prediction. When the predicted description was completely the same as the ground

[2]Here, we separated patterns regularly for each behavior, description, and box arrangement to appear uniformly in the training split. Thus, the model experienced all possible behaviors, descriptions, and box arrangements during training, but there were still 18 unexperienced episode patterns, which were "an unexperienced combination" of them.

truth, we judged it as correct. For this criterion, the model correctly described all 54 experienced patterns and the 18 unexperienced patterns.

Conversion from Language to Behavior

We next evaluated the model's capability to generate appropriate robot behaviors in response to an instruction depending on the current visual context. This conversion is carried out by utilizing the encoder of the l-EDM to embed an instructive sentence and then utilizing the decoder of the b-EDM to generate a behavioral sequence from the representation. First, we evaluated the performance based on the task execution by the real robot. When the robot moved the indicated box in the instructed direction more than 3 cm at the instructed speed, we judged the behavior generation to have been successful. Here, we regarded a generated behavior as SLOW if the robot took more than 30 time steps to return to the neighborhood of its initial posture; otherwise, we regarded it as FAST.

For this criterion, the model successfully generated a behavioral sequence for 36 out of the 54 experienced situations and for 12 out of the 18 unexperienced situations. Figure 6.3 visualizes examples of generated joint angle trajectories in failed cases. In these failed episodes, because the robot generated a trajectory slightly different from the target trajectory and it could not touch the correct side of the box, the robot failed in moving the box in the indicated direction. However, even in the failed cases, the model seems to have generated a trajectory that was rather similar to the predesigned target trajectory. Therefore, to evaluate the generated trajectories in more detail, we applied the dynamic time warping (DTW) [109] that measures similarities between temporal sequences. DTW analysis showed that even in failed cases, the generated joint angle sequence was most similar to the correct one of the 12 reference sequences. This indicates that the model at least learned the visually-conditioned relationships between linguistic expressions and corresponding robot behavioral sequences.

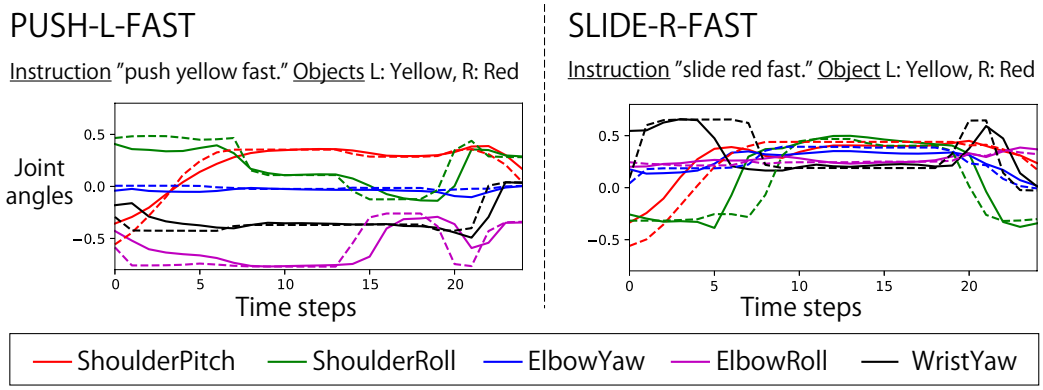

Figure 6.3. Examples in which the model failed to generate an appropriate behavior. The solid lines show the generated joint angles; the dashed lines represent the preprogrammed reference trajectories. We plotted only five joints on the moving arm. Even in the failed episodes, the model seems to have generated a joint angle sequence that was rather similar to the reference trajectory. In fact, DTW analysis demonstrated that in all the failed episodes, the generated sequence was most similar to the correct one among the 12 reference sequences.

6.5 Analysis of Shared Representations

Lastly, we investigated how the trained model internally represented the behavioral sequences and their corresponding sentences using PCA. The left panel in Figure 6.4 visualizes the embeddings of 18 possible descriptions. The plot shape, color, and fill represent the verb, object, and adverb, respectively. We can see that three parts of speech (namely verb, object, and adverb) were systematically embedded, i.e., the compositional structure of the descriptions was strongly represented in the PC1–PC2 plane.

The center panel of Figure 6.4 shows the representations of the behavioral sequences in the same PC1–2 plane. We can see that the behavioral sequences were actually bound with their paired sentences. Here, it is worth noting that the model could bind the same type of joint angle trajectory with different sentences depending on the current context. For instance, the point indicated as (A) is a representation of PUSH-R-SLOW in a context in which a yellow box and a red box were placed on the left and right, respectively; thus, the behavioral trajectory was bound with the sentence "push red slowly." On the other hand, the point indicated as (B) is also a representation of PUSH-R-SLOW in a context in which there were a green box and a yellow box on the left and right, respectively, and the behavior was hence bound with the sentence "push yellow slowly." Even

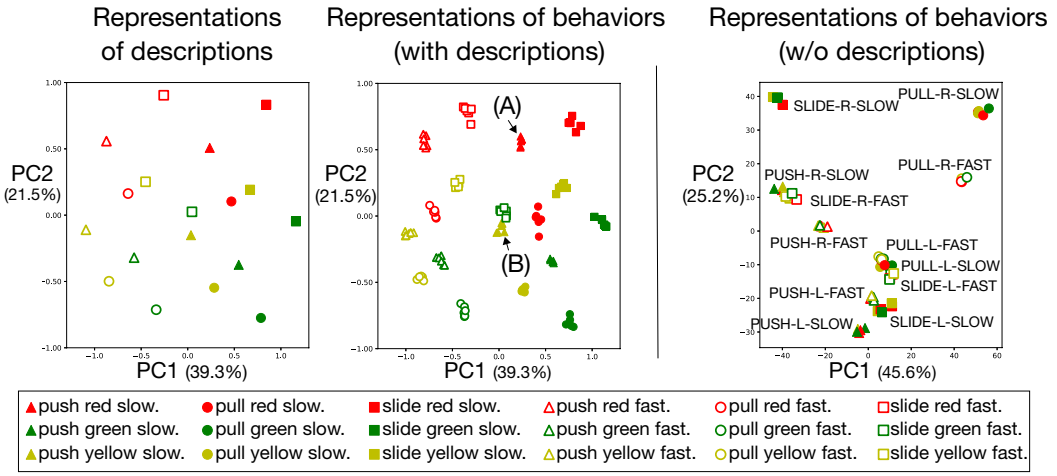

Figure 6.4. [Left] Representations of descriptions on the binding layer. [Center] Representations of behavioral sequences that were jointly learned with descriptions. They were bound with their descriptions, so they are represented in a semantic and compositional manner. [Right] Representations of behavioral sequences that were learned alone. In such a case, they were not organized in a semantic manner.

though points (A) and (B) encoded the same joint angle trajectory, they were far from each other because their meanings were different depending on the box combinations. On the other hand, different joint trajectories can be bound with the same sentence.

To verify that such semantic representations of behavior actually resulted from the joint learning with their paired sentences, we carried out another learning experiment in which we trained the b-EDM alone (i.e., the b-EDM was optimized by using the reconstruction loss only). This learning condition resulted in organizing the 12 clustered representations of the behavioral sequences that mainly encoded the joint angle information, as shown in the right panel of Figure 6.4. The visual context (i.e., box arrangement) was almost ignored, so it had almost no influence on the latent representations. This comparison suggests that the loss term to bind representations substantially influences the representations of behavioral sequences to be semantically grounded in their paired sentences.

6.6 Discussion

6.6.1 Comparison with Existing Bidirectional and Cross-modal Methods

This chapter proposed the two-coupled EDM architecture that bidirectionally converts between robot behavioral sequences and their descriptions. Existing studies have utilized EDMs mainly to convert sequences from a source domain to a target domain in a unidirectional and uni-modal (or cross-modal) manner. In contrast, the coupled EDM model proposed here learns to convert between behaviors and their descriptions bidirectionally from a paired language–behavior dataset. Furthermore, the proposed model has the potential capability of converting a behavioral sequence directly to another behavioral sequence that is semantically the same through the latent representation space, although we did not perform an evaluation of this capability. For instance, the PULL-R-SLOW behavior when a green box is put on the right side would be encoded as the semantic representation "pull green slowly". After that, if the green box was to be relocated on the left in the decoding phase, the model would decode the representation as the PULL-L-SLOW behavior.

Ogata et al. [41] proposed an NN architecture to bidirectionally convert between robot arm motions and their paired sentences. Their framework must iteratively run the BPTT algorithm to search an optimal latent vector with which to generate a sentence given a motion or vice versa. This takes a longer time and the convergence might not be stable. On the other hand, our proposed model only needs to run one forward-propagation to obtain a latent representation.

Plappert et al. [110] also built a framework to bidirectionally map human bodily motions into natural language descriptions and vice versa. Their framework was comprised of two independent unidirectional EDMs that were separately trained. This framework always requires a parallel dataset of bodily motions and textual annotations to train the model. On the other hand, the advantage of our method is that we can pretrain two RAEs separately using a uni-modal dataset and then retrain them with a parallel dataset by utilizing binding loss.

Noda et al. [111] proposed a time delay autoencoder model that learned to

extract low-dimensional features from a high-dimensional multimodal stream consisting of robot vision, audition, and proprioception. After learning, through the latent representation, the model could retrieve a lost modality from other modalities. For example, given only audition and proprioception streams, i.e., a blind situation, the model could imagine a visual situation accompanying them. However, their model has a limitation in terms of the capability to deal with temporal patterns because the length of time window is fixed. Therefore, temporal patterns longer than the window cannot be managed well.

6.6.2 Modeling Ambiguity Between Language and Behavior

The most important limitation of the current method is that it grounds a behavioral sequence in a sentence in a point-to-point manner. Essentially, we humans are able to express the same behavioral sequence in various ways as well as use the same expression for various behaviors. Even if the current context is apparently the same, the relations between language and behavior are still many-to-many depending on the participant's intent, the history of the discource, and so on. One promising way to cope with the intrinsic ambiguity of language would be the introduction of the Bayesian framework, similarly to the VAE [112]. By grounding language in behavior as a probabilistic distribution, we could explicitly model such many-to-many relationships. For example, there are some existing studies that proposed multimodal VAEs [87, 86]. These methods learn to model the probabilistic relationships between modalities in a generative form. Through the latent variable space, these models bidirectionally infer a data sample in one modality from the other modality. To date, these models have been focused on static data, such as images and binary attributes. In future work, by integrating such Bayesian schema into our RNN-based EDM framework, we could reasonably model the intrinsic ambiguity between language and behavior.

6.6.3 End-to-end Learning with Visual Feature Extractor

In the current experiment, we extracted visual features from the raw images using the pre-trained CAE in advance of the learning of the RAEs. This leads to the risk that the visual feature vectors achieved by the CAE are not optimal to the task because the CAE was trained with the objective function to reconstruct original images. In fact, we investigated the visual features extracted by the pre-trained CAE as a preliminary check. The analysis revealed that the lighting condition during data recording had a large influence on the visual feature space, although it did not have serious effects on the performance in the current task. One of the advantages of systems that wholly consist of NN architecture is that all modules are trained to be optimal to the imposed task in an end-to-end manner. In future work, to obtain more effective and noise-robust representations, learning should be carried out in an entirely end-to-end manner from raw image input to robot control output.

6.7 Summary

In this chapter, to attain bidirectional mapping between linguistic expressions and robot behavioral sequences, we proposed a two-coupled EDM framework in which the latent representations of behavioral sequences and their descriptions were bound with each other. The experiment demonstrated that this representation binding actually allows the model to convert behavioral sequences and their descriptions bidirectionally, conditioned on a visual context. The analysis of the latent representations revealed that behavioral sequences were represented in a semantic and compositional manner in the shared space by being learned together with their corresponding sentences. Future work will include an evaluation of the model's potential to learn more complicated tasks. We will also consider how to cope with the intrinsic ambiguity of language. Figure 6.5 summarizes the main targets discussed in this chapter.

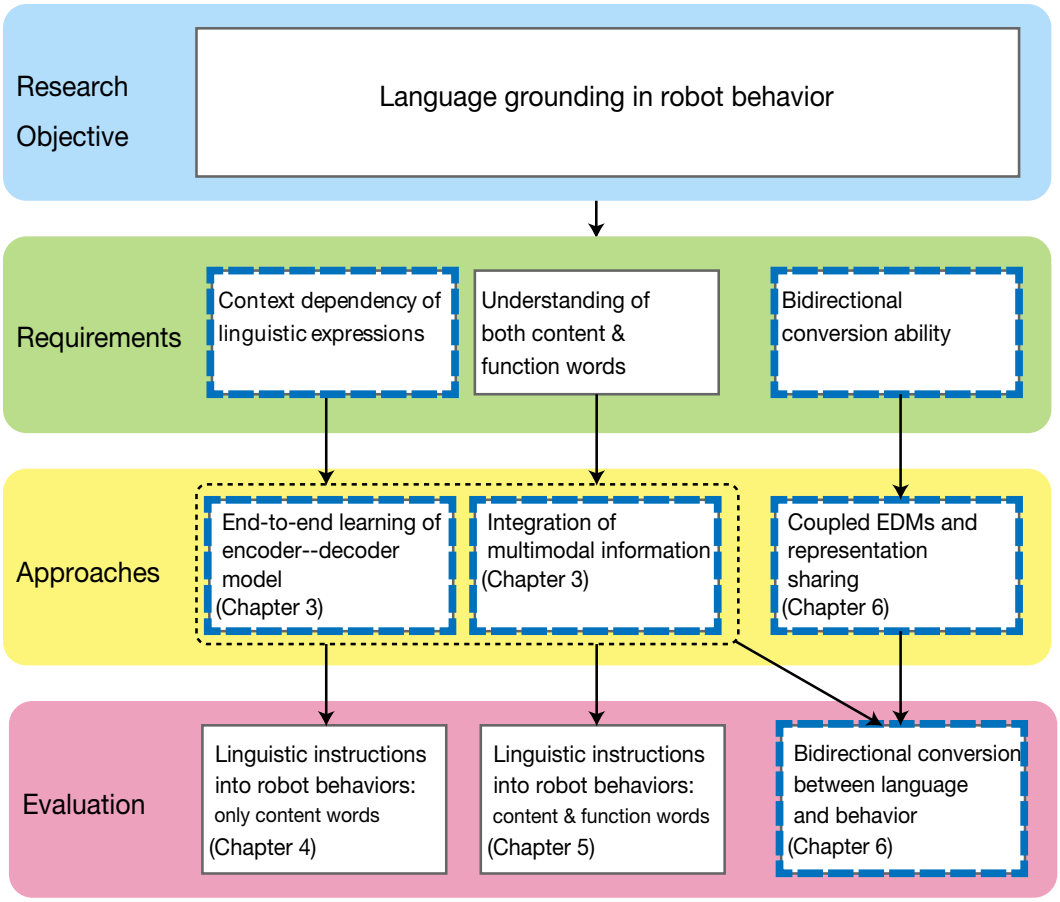

Figure 6.5. The main targets discussed in Chapter 6.

Chapter 7

Conclusion

7.1 Overall Summary of the Current Research

This study proposed a novel machine learning framework to ground language in robot behavior. We focused on three important issues: context dependency of mapping between language and behavior (R1), understanding of both content words and function words (R2), and bidirectional conversion ability (R3). To address R1 and R2, we proposed an RNN-based EDM that integrated multimodal information as input. It learned to ground linguistic instructions into robot behavioral sequences by integrating the instructions together with visual and proprioceptive information into the fixed-dimensional representation of its context state. R3 was addressed by combining two EDMs and introducing an additional loss function that binds the latent representations of language and behavior.

We evaluated the grounding capability of our proposed method with three robot experiments. In Chapter 4, we evaluated our framework on the first requirement. The task was to convert linguistic instructions that consist of only content words into robot behavioral sequences in an appropriate manner for the current visual context. The model successfully learned the given task. Specifically, the combinatorial relationships between color words and visual information were systematically learned, so the model could execute appropriate behavior even in unexperienced situations. The analysis of the internal representations actually revealed that the meanings of sentences were embedded in a way that was integrated with visual information. Moreover, the model represents not only grounding rela-

tionships but also the interaction progress pattern, which consists of instruction reception, behavior generation, and waiting as a cyclic attractor structure.

In Chapter 5, we evaluated the model on R2 by performing an experiment in which logic words were also included as examples of function words. Similarly to the first experiment, the F-EDM was able to learn the relationships between instructions and behavioral sequences with a certain level of compositional generalization. Color words were embedded by being integrated with visual information, and then they were grounded in left and right arm behaviors. Logic words were represented in accordance with their functions as logical operators. "Do" and "don't", which determine whether the model behaves as directed by a verb or in an inverse way, required the model to solve the X-OR problem implicitly included in the target data. The model actually extracted the problem and learned to resolve it in the relatively low-rank principal components by transforming input sentences non-linearly. The function of "or" that requires the robot to lift up either arm was organized as unstable areas of the RNN's internal dynamical system.

Chapter 6 performed the third experiment to evaluate the capability of bidirectionally converting between robot behavioral sequences and their linguistic expressions. By combining two EDMs and introducing the additional loss function that makes representations of language and corresponding behavior approach to each other, the paired sequences were actually embedded close to each other in the shared space. Through the shared representation, the bidirectional conversion was achieved, appropriately conditioned on the visual context.

7.2 Future Work

7.2.1 Scalability of the Proposed Framework

In future work, we should evaluate the scalability of the proposed learning framework. Although we performed robot experiments on a tabletop with some simple objects, practical situations are less constrained. Instructions given to robots also have much more various and complicated syntactic structure with a large vocabulary.

Recently, various types of deep NNs have achieved outstanding performance on mapping linguistic instructions to navigation command sequences in virtual environments [15, 17]. With great success in other kinds of grounding tasks, these examples lead us to expect the scalability of deep EDMs on language grounding in robot behavior. To scale up, improvements in model architecture and learning procedure would be required. Although this study adopted a supervised learning framework, this is not necessarily always suitable for target tasks. For example, in tasks that impose some goal-oriented behavior on a robot, even if all the collected supervising data succeeded in reaching the goal, there is a risk that the process or trajectory to reach the goal will be arbitrarily determined by a supervisor and is therefore not optimal. Also, these manually collected trajectories would not necessarily reflect the robot's body structure and dynamics.

In contrast, the successful cases of navigation in virtual environments often adopt a reinforcement learning framework. In reinforcement learning, robot behavior is explicitly optimized in relation to accomplishment of given tasks. In this sense, robots can autonomously develop their skills and achieve optimal behavior in the specific constraint of their own body. However, reinforcement learning has some issues when implemented in robots. First, rewards to be given to robots are not always clear. Secondly, reinforcement learning on robots takes much more time than supervised learning because it requires exhaustive exploration of a wide continuous space. The latter issue is serious because real robots can be broken and they can only work in real-time. To overcome these issues, some learning frameworks that combine supervised learning and reinforcement learning (and unsupervised learning) have been proposed [113]. There have also been methods that first pretrain a network in a simulated environment and then transfer the acquired knowledge to the real world [114]. We should evaluate the effectiveness of applying these methods to our framework.

7.2.2 Intrinsic Ambiguity of Language

In this study, we dealt with the context dependency, which is a highly important issue in language grounding. In the experiments performed here, the proposed model succeeded in appropriately converting between linguistic instructions and

robot behavioral sequences by perceiving the current context as a multimodal input stream. However, in practical situations, the mapping between natural language and the real world is more complicated. Beyond the context dependent change of relations between language and its referents, even if the current observation is apparently the same, there are infinite possible linguistic expressions for it. This would depend on the participant's intent, the history of the discourse, cultural background, etc. In contrast, there are also cases in which the meaning of an utterance cannot be uniquely determined only from the given context. Such intrinsic ambiguity of language cannot be completely solved. The symbol grounding problem is somewhat ill-posed. In fact, humans often make mistakes, but they are able to continue interaction by attempting to resolve them during interaction.

One way to address this inevitable ambiguity is to model the ambiguity between language and behavior explicitly. For instance, the VAE infers the posterior distribution of the latent variables given a data observation, in contrast to our proposed framework, which modeled the grounding relationships as points in a fixed-dimensional space. As another example, Brazinskas et al. [115] proposed a framework to embed words as probabilistic densities to model their potential meanings (e.g., "kiwi" could refer to either a bird or a fruit).

When we explicitly model the ambiguity or uncertainty of grounding relationships, we must also consider how to make a decision, which has been suspended once, from candidates. The first way is to ask the human partner for further clarity. Hatori et al. [116] built a deep learning system that inferred which object was referred to by natural language instructions from a human. If the human instruction had ambiguity, their model asked the human for clarification.

The other way is to start some actions with a tentative interpretation and to change the behavior whenever the tentative interpretation is wrong. For instance, Chen et al. [117] performed an experiment in which two humanoid robots controlled by an RNN interact with each other using a ball. Two robots knew several ball-playing patterns and these patterns were encoded as PB vectors. Here, PB vectors can be interpreted as a robot's intention that determines what kind of ball-playing pattern is generated. In this experiment, if a discrepancy between

two robot's intentions emerges during interaction, the generated behavioral sequences also become incompatible with each other. From the robots' view, this incompatibility is observed as an error between an RNN's own prediction and the perceived current state. In these situations, their model corrects PB values to minimize observed errors, i.e., to make the robot's intention compatible with the partner's intention through the current observation. By this function, the robots could continue interaction by flexibly changing their behavior among several ball-playing patterns. Implementing such an algorithm, robots would be able to dynamically understand language like humans. More precisely, they tentatively determine their plan in response to a linguistic instruction while simultaneously keeping its uncertainty or ambiguity, and then they adaptively change their behavior when they notice that they have misunderstood the instruction.

7.2.3 Social Aspects of Language

This study does not deal with social aspects of language. In practice, our language systems are used in situations of mutual intertwined communication. One person's utterance or behavior influences their partner's responding utterance or behavior and vice versa. This can be further expanded situations with more people involved. Explicitly modeling other agents' internal state or intention during interaction (e.g., refer to Rabinowitz et al. [118]) would be required to correctly infer the current social context. Also, the grounding relationships would be more or less different among various communities (e.g., families, regions, generations, and cultures). Moreover, in such interaction, the relationships between linguistic expressions and their referents would be dynamically changing. Therefore, the skill to promptly adapt language use online is also required [119].

It has also been reported that social cognition is important during the language acquisition process. Infants can efficiently acquire their first language scaffolded from their caregivers' support [47]. Recently, the importance of curriculum learning has also been demonstrated in some machine learning experiments [120, 121, 15]. By adopting these techniques, more efficient learning of language grounding in robot behavior can be achieved.

7.3 Significance of the Current Study in Embodiment Informatics

Creating service robots that work by understand natural language is a highly important theme nowadays. Up to now, language has been mainly dealt with in the field of informatics, but robots have been developed in the field of mechanical engineering. Because the mathematical methods that express them have been different from each other, we must consider how to bridge the gap between language and robots. As the symbol grounding problem states, it is difficult to manually design a good interface between high-level intelligence that processes abstract symbolic representations and low-level intelligence that deals with raw sensorimotor flow coupled with the dynamic continuous world.

In contrast to the top-down method in which humans arbitrarily design interfaces, learning-based approaches in which grounding relations are learned from systems' own experiences in a bottom-up manner is becoming a major concept, as seen in the recent trend of deep learning. It has been reported that NNs can gradually connect from raw-level sensorimotor features to their abstract symbolic representations through deeply stacked layers. The current study also demonstrated that an RNN-based system can learn to encode the different modalities of language, vision, and robot proprioception together into an integrated representation and bidirectionally convert language and robot behavior through such a representation.

When we attempt to build robot intelligence by learning, one of the key concepts is "*embodiment*." The idea of embodied intelligence hypothesizes that our intelligent cognition, including symbol use, is developed from our own embodied experiences of the world. Based on this philosophy, robots can also develop their own cognitive functions from their own experiences in a manner specific to their own body structure and dynamics. Specifically, functions that are closely coupled with their physical characteristics, such as locomotion, reaching, and grasping, can benefit greatly from the concept of embodiment.

Here, the problem is whether linguistic skill is also acquired sufficiently from only bottom-up learning. The point is that language is a tool for communica-

tion with other agents, and robots must also communicate with humans using language. In the case of communication between humans, we have rather similar bodies and a shared structure of experiences; therefore, we develop relatively similar grounding structure. However, the grounding form developed by robots, which have much different experiences from humans, might be very different from ours. In fact, Mordatch and Abbeel [122] focused on symbol emergence between multi-agents in a virtual environment and performed an experiment in which agents learned to execute a task in a collaborative manner by communicating with each other using symbolic signals. They reported that the agents sometimes developed a symbol system that could not be easily understood by humans, but they were still effective to achieve their goal. Besides the example of a multi-agent system, it has also been reported that deep NNs can outperform humans in image classification tasks, but they can make mistakes for images that humans never fail to recognize[1]. As pointed out, NNs are basically black boxes; even if they seemingly have the same function as humans, the internal representation and process can be different.

Considering these facts, although learning-based approaches make a significant contribution to robot intelligence, robots must achieve language skills in a manner that is not only embodied in their experiences but also able to be shared with humans. Therefore, although we do not have to perfectly replicate our own cognitive process to implement it on robots, it is important to investigate how robots internally represent relations between language and their own behavior, as in the current study. In future, to design fields in which humans and robots, which are different entities, richly interact with each other, we must consider how the whole process of interaction dynamically continues. In Peircian terminology, it is not enough to model the static relations between input (sign) and output (object); we must consider how the robot (interpretant) internally understands the sign by dynamically grounding it in every given context and how different agents build a dynamical process of mutual communication and understanding.

The investigation we performed in the current study to understand the internal representation of the link between language and robot behavior in NNs

[1]There have also been studies that attempted to make such adversarial examples [123].

contributes the construction of novel styles of interaction between heterogeneous agents (human–human, human–robot, and robot–robot) beyond forcing our naive perspective on conventional language use. In that sense, we believe that the current study makes a meaningful contribution to the field of embodiment informatics.

7.4 Significance of the Current Study in Intermedia Art and Science

With the development of sensing techniques and communication technology, it is important to consider how to integrate and utilize sensed multimodal information. We think that the objective of the intermedia art and science research is roughly divided into (1) developing scientific knowledge on integration of multimodal information, (2) applying the knowledge into engineering products, and (3) creating novel values or perspectives, which would be impossible only with conventional technology, as art. The first two objectives overlap with the embodiment informatics. The research on robots, which consist of densely arranged sensorimotor modules and intelligent modules to integrate multimodal information, is on the cutting-edge of the engineering side of the intermedia art and science. We have already explained the importance of the methodology that investigates how the intelligent systems internally represent multimodal information.

However, we think that this methodology can also be applied to the art side. Creation of a new style of expressions is inseparable with building a novel representation of the world. In recent, generative NNs, such as VAEs and generative adversarial networks [100], have been used to model various kinds of data distributions including images, videos, pieces of music, and so on. By exploring the latent representations of them, we can produce novel pieces. Because these frameworks can be applied to multimodal data, we can expect that in literal, novel "intermedia arts" are produced. Utilization of different bodies (e.g., robots) would make novel representation spaces, which origin from experiences embodied differently from us, therefore lead to artistic expressions that make us be aware of new perspective of the world. Moreover, the integration of linguistic knowledge

would give a useful interface to a wide range of creators and make opportunities of collaborative work between humans and systems.

Considering these future directions, we believe that our methodology, which build integrated representations of multimodal information on robots in a constructive manner, is one of fundamental approaches of intermedia art and science research.

Bibliography

[1] S. Harnad, "The Symbol Grounding Problem," *Physica D:*, vol. 42, no. 1-3, pp. 335–346, 1990.

[2] A. Nguyen, D. Kanoulas, L. Muratore, D. G. Caldwell, and N. G. Tsagarakis, "Translating Videos to Commands for Robotic Manipulation with Deep Recurrent Neural Networks," in *2018 IEEE International Conference on Robotics and Automation (ICRA2018)*, (Brisbane, Australia), 2018.

[3] Y. Pan, T. Yao, H. Li, and T. Mei, "Video captioning with transferred semantic attributes," in *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR2017)*, (Hawaii, USA), 2017.

[4] A. Cangelosi, G. Metta, G. Sagerer, S. Nolfi, C. Nehaniv, K. Fischer, J. Tani, T. Belpaeme, G. Sandini, F. Nori, L. Fadiga, B. Wrede, K. Rohlfing, E. Tuci, K. Dautenhahn, J. Saunders, and A. Zeschel, "Integration of Action and Language Knowledge : A Roadmap for Developmental Robotics," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 3, pp. 167–195, 2010.

[5] T. Taniguchi, T. Nagai, T. Nakamura, N. Iwahashi, T. Ogata, and H. Asoh, "Symbol Emergence in Robotics: A Survey," *Advanced Robotics in press*, vol. 30, no. 11-12, pp. 706–728, 2016.

[6] 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, vol. 1, pp. 12–34, may 2009.

[7] R. Pfeifer and C. Scheier, *Understanding Intelligence*. MIT press, 2001.

[8] A. Cangelosi, M. Schlesinger, and L. B. Smith, *Developmental robotics : from babies to robots*. MIT press, 2015.

[9] Y. Sugita and J. Tani, "Learning Semantic Combinatoriality from the Interaction between Linguistic and Behavioral Processes," *Adaptive Behavior*, vol. 13, no. 1, pp. 33–52, 2005.

[10] E. Tuci, T. Ferrauto, A. Zeschel, G. Massera, and S. Nolfi, "An experiment on behavior generalization and the emergence of linguistic compositionality in evolving robots," *IEEE Transactions on Autonomous Mental Development*, vol. 3, no. 2, pp. 176–189, 2011.

[11] F. Stramandinoli, D. Marocco, and A. Cangelosi, "Making sense of words : a robotic model for language abstraction," *Autonomous Robots*, vol. 41, no. 2, pp. 367–383, 2017.

[12] B. H. Partee, *Compositionality in Formal Semantics: selected papers*. John Wiley & Sons, 2003.

[13] M. R. D'Amato, D. P. Salmon, E. Loukas, and A. Tomie, "Symmetry and Transitivity of Conditional Relations in Monkeys (Cebus Apella) and Pigeons (Columba Livia)," *Journal of the Experimental Analysis of Behavior*, vol. 44, pp. 35–47, 1985.

[14] M. Sidman, "Symmetry and Equivalence Relations in Behavior," *Cognitive Studies*, vol. 15, no. 3, pp. 322–332, 2008.

[15] K. M. Hermann, F. Hill, S. Green, F. Wang, R. Faulkner, H. Soyer, D. Szepesvari, W. M. Czarnecki, M. Jaderberg, D. Teplyashin, M. Wainwright, C. Apps, D. Hassabis, and P. Blunsom, "Grounded Language Learning in a Simulated 3D World," *arXiv preprint*, no. 1706.06551, pp. 1–22, 2017.

[16] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov, "Gated-Attention Architectures for Task-Oriented Language Grounding," in *The 32nd AAAI Conference on Artificial Intelligence (AAAI2018)*, (Louisiana, USA), 2018.

[17] P. Anderson, D. Teney, J. Bruce, M. Johnson, S. Niko, and I. Reid, "Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Utah, USA), 2018.

[18] H. Yu, H. Zhang, and W. Xu, "A Deep Compositional Framework for Human-like Language Acquisition in Virtual Environment," *arXiv preprint*, no. 1703.09831, pp. 1–16, 2017.

[19] D. Misra, J. Langford, and Y. Artzi, "Mapping Instructions and Visual Observations to Actions with Reinforcement Learning," in *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, (Copenhagen, Denmark), 2017.

[20] Y. Aytar, C. Vondrick, and A. Torralba, "See, Hear, and Read: Deep Aligned Representations," *arXiv preprint*, no. 1706.00932, 2017.

[21] A. Duarte, D. Surís, A. Salvador, and X. Giró, "Temporal-aware Cross-modal Embeddings for Video and Audio Retrieval," in *31st Conference on Neural Information Processing Systems (NIPS 2017), workshop*, (CA, USA), 2017.

[22] R. Arandjelović and A. Zisserman, "Objects that Sound," in *2018 European Conference on Computer Vision*, (Munich, Germany), 2018.

[23] J. L. Elman, "Finding Structure in Time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[24] M. Jordan, "Serial order: A parallel distributed processing approach," *Advances in Psychology*, vol. 121, pp. 471–495, 1997.

[25] I. Sutskever, O. Vinyals, and V. Q. Le, "Sequence to Sequence Learning with Neural Networks," in *Neural Information Processing Systems 2014 (NIPS2014)*, (Montreal, Canada), 2014.

[26] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *IEEE International Conference on Learning Representations (ICLR2015)*, (CA, USA), 2015.

[27] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's Neural Machine Translation System : Bridging the Gap between Human and Machine Translation," *arXiv preprint*, no. 1609.08144, 2016.

[28] O. Vinyals and V. Q. Le, "A Neural Conversational Model," in *the 32nd International Conference on Machine Learning (ICML2015) workshop*, (Lille, France), 2015.

[29] S. Dave, J. Parikh, and P. Bhattacharyya, "Interlingua-based English Hindi Machine Translation and Language Divergence," *Machine Translation*, vol. 16, no. 4, pp. 251–304, 2001.

[30] P. Koehn, F. J. Och, and D. Marcu, "Statistical Phrase-Based Translation," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, vol. 1, (Edmonton, Canada), pp. 48–54, 2003.

[31] A. Newell and H. a. Simon, "Computer Science as Empirical Inquiry: Symbols and Search," *Communications of the ACM*, vol. 19, no. 3, pp. 113–126, 1976.

[32] A. Newell, "Physical symbol systems," *Cognitive Science*, vol. 4, no. 2, pp. 135–183, 1980.

[33] J. Weizenbaum, "ELIZA — A Computer Program For the Study of Natural Language Communication Between Man And Machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.

[34] T. Winograd, "Procedures as a representation for data in a computer program for understanding natural language," *MIT AI Technical Report*, vol. 235, 1971.

[35] S. H. Liao, "Expert system methodologies and applications-a decade review from 1995 to 2004," *Expert Systems with Applications*, vol. 28, no. 1, pp. 93–103, 2005.

[36] M. Campbell, a. J. Hoane Jr., and F.-h. Hsu, "Deep Blue," *Artificial Intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002.

[37] J. R. Searle, "Minds, brains, and programs," *Behavioral and Brain Sciences*, vol. 3, no. 3, pp. 417–424, 1980.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances In Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[39] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and Tell: A Neural Image Caption Generator," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015)*, (Massachusetts, USA), nov 2015.

[40] J. Nishihara, T. Nakamura, and T. Nagai, "Online Algorithm for Robots to Learn Object Concepts and Language Model," *Transactions on Cognitive and Developmental Systems*, vol. 9, no. 3, pp. 255–268, 2017.

[41] T. Ogata, M. Murase, J. Tani, K. Komatani, and H. G. Okuno, "Two-way Translation of Compound Sentences and Arm Motions by Recurrent Neural Networks," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, (San Diego), pp. 1858–1863, IEEE, 2007.

[42] M. Polanyi, *The Tacit Dimension*. Garden City, N.Y., Doubleday, 1966.

[43] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick, "Neuroscience-Inspired Artificial Intelligence," *Neuron*, vol. 95, no. 2, pp. 245–258, 2017.

[44] A. F. Morse, V. L. Benitez, T. Belpaeme, A. Cangelosi, and L. B. Smith, "Posture affects how robots and infants map words to objects," *PLoS ONE*, vol. 10, no. 3, pp. 5–8, 2015.

[45] T. Nakamura, T. Nagai, K. Funakoshi, S. Nagasaka, T. Taniguchi, and N. Iwahashi, "Mutual learning of an object concept and language model based on MLDA and NPYLM," in *2014 IEEE International Conference on Intelligent Robots and Systems (IROS2014)*, (IL, USA), pp. 600–607, 2014.

[46] T. Taniguchi, R. Yoshino, and T. Takano, "Multimodal hierarchical dirichlet process-based active perception by a robot," *Frontiers in Neurorobotics*, vol. 12, no. 22, pp. 1–19, 2018, 1510.00331.

[47] M. Tomasello, *Constructing a language: A usage-based theory of language acquisition*. Harvard University Press, Cambridge, 2003.

[48] N. Chomsky, *Syntactic Structures*. 's-Gravenhage: Mouton & Co., 1957.

[49] L. Wittgenstein and J. Schulte, *Philosophical Investigations*. Macmillan, 1953.

[50] O. Hauk, I. Johnsrude, and F. Pulvermu, "Somatotopic Representation of Action Words in Human Motor and Premotor Cortex," *Neuron*, vol. 41, no. 2, pp. 301–307, 2004.

[51] T. Inamura, I. Toshima, H. Tanie, and Y. Nakamura, "Embodied Symbol Emergence Based on Mimesis Theory," *The International Journal of Robotics Research*, vol. 23, no. 4, pp. 363–377, 2004.

[52] W. Takano and Y. Nakamura, "Statistically integrated semiotics that enables mutual inference between linguistic and behavioral symbols for humanoid robots," in *2009 IEEE International Conference on Robotics and Automation*, (Kobe, Japan), pp. 646–652, 2009.

[53] T. Nakamura, T. Nagai, and N. Iwahashi, "Multimodal categorization by hierarchical Dirichlet process," in *2011 IEEE International Conference on Intelligent Robots and Systems*, (CA, USA), pp. 1520–1525, IEEE, 2011.

[54] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy, "Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation," in *Proceedings of 25th AAAI Conference on Artificial Intelligence*, (CA, USA), pp. 1507–1514, 2011.

[55] T. Araki, T. Nakamura, T. Nagai, S. Nagasaka, T. Taniguchi, and N. Iwahashi, "Online Learning of Concepts and Words Using Multimodal LDA and Hierarchical Pitman-Yor Language Model," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Vilamoura, Portugal), pp. 1623–1630, IEEE, 2012.

[56] M. Fadlil, K. Ikeda, K. Abe, T. Nakamura, and T. Nagai, "Integrated concept of objects and human motions based on multi-layered multimodal LDA," in *IEEE International Conference on Intelligent Robots and Systems*, (Tokyo, Japan), pp. 2256–2263, 2013.

[57] M. Attamimi, M. Fadlil, K. Abe, T. Nakamura, K. Funakoshi, and T. Nagai, "Integration of various concepts and grounding of word meanings using multi-layered multimodal LDA for sentence generation," in *IEEE International Conference on Intelligent Robots and Systems*, (IL, USA), pp. 2194–2201, 2014.

[58] A. Taniguchi, T. Taniguchi, and T. Inamura, "Spatial Concept Acquisition for a Mobile Robot That Integrates Self-Localization and Unsupervised Word Discovery From Spoken Sentences," *IEEE Transaction on Cognitive and Developmental Systems*, vol. 8, no. 4, pp. 285–297, 2016.

[59] W. Takano, S. Hamano, and Y. Nakamura, "Correlated space formation for human whole-body motion primitives and descriptive word labels," *Robotics and Autonomous Systems*, vol. 66, pp. 35–43, 2015.

[60] A. Cangelosi and T. Riga, "An embodied model for sensorimotor grounding and grounding transfer: Experiments with epigenetic robots," *Cognitive Science*, vol. 30, no. 4, pp. 673–689, 2006.

[61] H. Arie, T. Endo, S. Jeong, M. Lee, S. Sugano, and J. Tani, "Integrative learning between language and action: A neuro-robotics experiment," in *20th International Conference on Artificial Neural Networks (ICANN2010)*, (Thessaloniki,Greece), pp. 256–265, 2010.

[62] D. Marocco, A. Cangelosi, K. Fischer, and T. Belpaeme, "Grounding action words in the sensorimotor interaction with the world: Experiments with a simulated icub humanoid robot," *Frontiers in Neurorobotics*, vol. 4, no. 7, pp. 1–15, 2010.

[63] X. Hinaut, M. Petit, G. Pointeau, and P. F. Dominey, "Exploring the acquisition and production of grammatical constructions through human-robot interaction with echo state networks.," *Frontiers in neurorobotics*, vol. 8, no. 16, 2014.

[64] S. Heinrich and S. Wermter, "Interactive Language Understanding with Multiple Timescale Recurrent Neural Networks," *Artificial Neural Networks and Machine Learning — ICANN 2014, Lecture Notes in Computer Science (LNCS)*, vol. 8681, pp. 193–200, 2014.

[65] H. Jaeger and H. Haas, "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[66] J. Tani and N. Fukumura, "Embedding a grammatical description in deterministic chaos: an experiment in recurrent neural learning," *Biological Cybernetics*, vol. 72, pp. 365–370, 1995.

[67] J. Namikawa, R. Nishimoto, and J. Tani, "A Neurodynamic Account of Spontaneous Behaviour," *PLoS computational biology*, vol. 7, no. 10, 2011.

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

[69] F. A. Gers and J. Schmidhuber, "Recurrent Nets that Time and Count," *Proc. of the IEEE-INNS-ENNS Int. Joint Conf. on Neural Networks*, vol. 3, pp. 189–194, 2000.

[70] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, pp. 1–46, 2015.

[71] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "SmoothGrad: removing noise by adding noise," *arXiv preprint*, no. 1706.03825, 2017.

[72] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN : Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, pp. 2172–2180, 2016.

[73] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention," in *International Conference on Machine Learning*, (Lille, France), pp. 2048–2057, 2015.

[74] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep Speech: Scaling up end-to-end speech recognition," *arXiv preprint*, no. 1412.5567, pp. 1–12, 2014.

[75] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dimitriadis, X. Cui, B. Ramabhadran, M. Picheny, L. L. Lim, B. Roomi, and P. Hall, "English conversational telephone speech recognition by humans and machines," in *Proceedings of the Annual Conference of the International Speech Communication Association, Interspeech*, (Stockholm, Sweden), pp. 132–136, 2017.

[76] S. Fan, Y. Qian, and F. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in *Proceedings of the Annual Conference of the International Speech Communication Association, Interspeech*, (Singapore), pp. 1964–1968, 2014.

[77] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis, "Parallel WaveNet: Fast High-Fidelity Speech Synthesis," *arXiv preprint*, no. 1711.10433, 2017.

[78] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent Trends in Deep Learning Based Natural Language Processing," *arXiv preprint*, no. 1708.02709, pp. 1–22, 2017.

[79] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[80] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[81] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[82] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Int.Conf. on Learning Representations (ICLR2015)*, (CA, USA), 2015.

[83] H.-Y. F. Tung, A. W. Harley, L.-K. Huang, and K. Fragkiadaki, "Reward Learning from Narrated Demonstrations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2018)*, (Utah, USA), pp. 7004–7013, 2018, 1804.10692.

[84] K. Suzuki, H. Mori, and T. Ogata, "Motion Switching With Sensory and Instruction Signals by Designing Dynamical Systems Using Deep Neural Network," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3481–3488, 2018.

[85] E. Hughes, F. Hill, J. Leike, P. Kohli, and E. Grefenstette, "Learning to Follow Language Instructions with Adversarial Reward Induction," *arXiv preprint*, no. 1806.01946, 2018.

[86] M. Suzuki, K. Nakayama, and Y. Matsuo, "Joint Multimodal Learning with Deep Generative Models," in *International Conference on Learning Representations (ICLR2017), workshop*, (Toulon, France), pp. 1–12, 2017.

[87] M. Wu and N. Goodman, "Multimodal Generative Models for Scalable Weakly-Supervised Learning," in *Neural Information Processing Systems 2018 (NeurIPS2018)*, (Montreal, Canada), 2018.

[88] K. J. Lang, A. H. Waibel, and G. E. Hinton, "A time-delay neural network architecture for isolated word recognition," *Neural Networks*, vol. 3, no. 1, pp. 23–43, 1990.

[89] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.

[90] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pp. 318–362, MIT Press, 1986.

[91] Y. Yamashita and J. Tani, "Emergence of Functional Hierarchy in a Multiple Timescale Neural Network Model: a Humanoid Robot Experiment.," *PLoS Computational Biology*, vol. 4, no. 11, pp. 1–18, 2008.

[92] W. Hinoshita, H. Arie, J. Tani, H. G. Okuno, and T. Ogata, "Emergence of hierarchical structure mirroring linguistic composition in a recurrent neural network.," *Neural Networks*, vol. 24, no. 4, pp. 311–320, 2011.

[93] D. Ha and J. Schmidhuber, "World Models," in *Neural Information Processing Systems 2018 (NeurIPS2018)*, (Montreal, Canada), 2018.

[94] A. Graves, "Generating Sequences With Recurrent Neural Networks," *arXiv preprint*, no. 1308.0850, pp. 1–43, 2013.

[95] S. Murata, Y. Yamashita, H. Arie, T. Ogata, S. Sugano, and J. 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*, vol. 28, no. 4, pp. 830–848, 2017.

[96] J. Namikawa and J. Tani, "Learning to imitate stochastic time series in a compositional way by chaos," *Neural Networks*, vol. 23, no. 5, pp. 625–638, 2010.

[97] F. Saussure, *Course in General Linguistics*. New York: Philosophical Library, 1959.

[98] S. Heinrich, S. Magg, and S. Wermter, "Analysing the Multiple Timescale Recurrent Neural Network for Embodied Language Understanding," in *Artificial Neural Networks*, pp. 149–174, Springer Cham, 2015.

[99] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of NAACL-HLT 2013*, (Atlanta), pp. 746–751, 2013.

[100] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," in *International Conference on Learning Representations (ICLR2016)*, (San Juan, Puerto Rico), 2016.

[101] D. Ha and D. Eck, "A Neural Representation of Sketch Drawings," in *International Conference on Learning Representations (ICLR2018)*, (Vancouver, Canada), pp. 1–16, 2018.

[102] J. Li, X. Chen, E. Hovy, and D. Jurafsky, "Visualizing and Understanding Neural Models in NLP," in *The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (CA, USA), 2016.

[103] B. M. Lake and M. Baroni, "Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks," in *the 35th International Conference on Machine Learning (ICML 2018)*, (Stockholm, Sweden), 2018.

[104] A. Liška, G. Kruszewski, and M. Baroni, "Memorize or generalize? Searching for a compositional RNN in a haystack," in *the 35th International Conference on Machine Learning (ICML 2018), AEGAP workshop*, (Stockholm, Sweden), 2018.

[105] J. Loula, M. Baroni, and B. M. Lake, "Rearranging the Familiar: Testing Compositional Generalization in Recurrent Networks," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, (Brussels, Belgium), pp. 108–114, 2018.

[106] F. Förster, C. L. Nehaniv, and J. Saunders, "Robots that say 'no'," in *Advances in Artificial Life: Darwin Meets Von Neumann*, pp. 158–166, Springer-Verlag, 2011.

[107] O. Fabius and J. R. V. Amersfoort, "Variational Recurrent Auto-encoders," in *ICLR2015 workshop*, (CA, USA), 2015.

[108] G. E. Hinton and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[109] S. Salvador and P. Chan, "FastDTW : Toward Accurate Dynamic Time Warping in Linear Time and Space," *Intelligent Data Analysis*, vol. 11, pp. 561–580, 2007.

[110] M. Plappert, C. Mandery, and T. Asfour, "Learning a bidirectional mapping between human whole-body motion and natural language using deep recurrent neural networks," *Robotics and Autonomous Systems*, vol. 109, pp. 13–26, 2018.

[111] K. Noda, H. Arie, Y. Suga, and T. Ogata, "Multimodal integration learning of robot behavior using deep neural networks," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 721–736, 2014.

[112] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *International Conference on Learning Representations (ICLR2014)*, (Banff, Canada), 2014.

[113] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[114] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-Real Robot Learning from Pixels with Progressive Nets," in *The 1st Conference on Robot Learning (CoRL 2017)*, (CA, USA), 2017.

[115] A. Bražinskas, S. Havrylov, and I. Titov, "Embedding Words as Distributions with a Bayesian Skip-gram Model," in *The 27th International Conference on Computational Linguistics (COLING 2018)*, (New Mexico, USA), 2018.

[116] J. Hatori, Y. Kikuchi, S. Kobayashi, and K. Takahashi, "Interactively Picking Real-World Objects with Unconstrained Spoken Language Instructions," in *2018 IEEE Int. Conf. on Robotics and Automation (ICRA2018)*, (Brisbane, Australia), 2018.

[117] Y. Chen, S. Murata, H. Arie, T. Ogata, J. Tani, and S. Sugano, "Emergence of Interactive Behaviors between Two Robots by Prediction Error

Minimization Mechanism," in *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, (Paris, France), pp. 302–307, 2016.

[118] N. C. Rabinowitz, F. Perbet, H. F. Song, C. Zhang, S. M. A. Eslami, and M. Botvinick, "Machine Theory of Mind," in *Proceedings of the 35th International Conference on Machine Learning (ICML2018)*, (Stockholm, Sweden), pp. 4218–4227, 2018.

[119] A. Lerer and A. Peysakhovich, "Learning Social Conventions in Markov Games," *arXiv preprint*, no. 1806.10071, 2018.

[120] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio, "BabyAI: First Steps Towards Grounded Language Learning With a Human In the Loop," *arXiv preprint*, no. 1810.08272, pp. 1–13, 2018.

[121] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, and P. Blunsom, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[122] I. Mordatch and P. Abbeel, "Emergence of Grounded Compositional Language in Multi-Agent Populations," in *The 32nd AAAI Conference on Artificial Intelligence*, (Louisiana, USA), 2018.

[123] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations (ICLR2014)*, (Banff, Canada), 2014.

Appendix A

Details of Neural Networks

A.1 Multi Timescale Recurrent Neural Network

Here, we describe the forward dynamics of an L-layers MTRNN. The internal state vector of the lth context layer at time step t, namely $\boldsymbol{c}_t^{(l)}$, is computed by the following equation:

$$\boldsymbol{c}_t^{(l)} = \begin{cases} \left(1-\frac{1}{\tau_l}\right)\boldsymbol{c}_{t-1}^{(l)} + \frac{1}{\tau_l}\left(\boldsymbol{W}^{\text{in}}\boldsymbol{x}_t + \boldsymbol{W}^{(l,l)}\boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{W}^{(l+1,l)}\boldsymbol{h}_{t-1}^{(l+1)} + \boldsymbol{b}^{(l)}\right) & (l=1), \\ \left(1-\frac{1}{\tau_l}\right)\boldsymbol{c}_{t-1}^{(l)} + \frac{1}{\tau_l}\left(\boldsymbol{W}^{(l-1,l)}\boldsymbol{h}_{t-1}^{(l-1)} + \boldsymbol{W}^{(l,l)}\boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{b}^{(l)}\right) & (l=L), \\ \left(1-\frac{1}{\tau_l}\right)\boldsymbol{c}_{t-1}^{(l)} + \frac{1}{\tau_l}\left(\boldsymbol{W}^{(l-1,l)}\boldsymbol{h}_{t-1}^{(l-1)} + \boldsymbol{W}^{(l,l)}\boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{W}^{(l+1,l)}\boldsymbol{h}_{t-1}^{(l+1)} + \boldsymbol{b}^{(l)}\right) & (\text{otherwise}), \end{cases} \tag{A.1}$$

where τ_l is a time constant that controls the response speed of the nodes in the lth layer, $\boldsymbol{W}^{\text{in}}$ is a weight matrix from the input layer to the 1st layer, $\boldsymbol{W}^{(i,j)}$ is a weight matrix from the ith layer to the jth layer, and $\boldsymbol{b}^{(l)}$ is a bias term on the lth layer. $\boldsymbol{h}_t^{(l)}$ is calculated by applying an activation function tanh to $\boldsymbol{c}_t^{(l)}$. After these calculations, the activation of the first layer $\boldsymbol{h}_t^{(1)}$ is projected to the output layer as follows:

$$\boldsymbol{y}_t = \tanh(\boldsymbol{W}^{\text{out}}\boldsymbol{h}_t^{(1)} + \boldsymbol{b}^{\text{out}}). \tag{A.2}$$

A.2 Stochastic Modeling of Temporal Data for Stable Learning

One of most commonly used loss functions to train neural networks by the gradient descent is MSE. This is equivalent to maximizing the likelihood by assuming that data points are always given noise from the same Gaussian distribution. However, the actual magnitude of variance or uncertainty in real time-series data can depend on the context. In these cases, i.e., when some parts of data have larger noise or unpredictability than other parts, usage of MSE as the loss function might make learning unstable because the RNN struggles to learn these parts.

To avoid this issue, Murata et al. [95] introduced another loss function. This loss function assumes that each data point is added Gaussian noise whose variance depends on the history. With this loss function, the RNN learns to predict not only the external state in the future but also its uncertainty as variance. Thanks to such a loss, which presumes the context dependent unpredictability in target data, the errors back-propagated to the trainable parameters decrease with respect to such unpredictable parts by correctly predicting their uncertainty. Therefore, the RNN can stably learn temporal patterns included in target data. To predict variance, the variance layer is added as follows:

$$\boldsymbol{v}_t = \exp(\boldsymbol{W}^{\mathrm{var}}\boldsymbol{h}_t^{(1)} + \boldsymbol{b}^{\mathrm{var}}). \tag{A.3}$$

Since variance must be positive, the exp function is used for activation. Here, target data is denoted as $\hat{y}_t$, and the likelihood function L_t is expressed as follows:

$$L_t = \prod_i \frac{1}{\sqrt{2\pi v_{t,i}}} \exp\left(-\frac{(\hat{y}_{t,i} - y_{t,i})^2}{2v_{t,i}}\right), \tag{A.4}$$

where i is the index for output and variance nodes. The network is trained by maximizing this function. To formulate the problem as a minimization problem and to simplify the problem, we instead use the negative log likelihood function $-\log L$.

A.3 Long Short-Term Memory Unit

The behavior of an LSTM layer at time step t is calculated as follows:

$$\begin{aligned}
\boldsymbol{a}_t &= \tanh(\boldsymbol{W}_{\mathrm{IN}}\boldsymbol{x}_t + \boldsymbol{R}_{\mathrm{IN}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\mathrm{IN}}), && \text{(A.5)}\\
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_{\mathrm{IG}}\boldsymbol{x}_t + \boldsymbol{R}_{\mathrm{IG}}\boldsymbol{h}_{t-1} + \boldsymbol{p}_{\mathrm{IG}} \odot \boldsymbol{c}_{t-1} + \boldsymbol{b}_{\mathrm{IG}}), && \text{(A.6)}\\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{\mathrm{FG}}\boldsymbol{x}_t + \boldsymbol{R}_{\mathrm{FG}}\boldsymbol{h}_{t-1} + \boldsymbol{p}_{\mathrm{FG}} \odot \boldsymbol{c}_{t-1} + \boldsymbol{b}_{\mathrm{FG}}), && \text{(A.7)}\\
\boldsymbol{c}_t &= \boldsymbol{i}_t \odot \boldsymbol{a}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1}, && \text{(A.8)}\\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_{\mathrm{OG}}\boldsymbol{x}_t + \boldsymbol{R}_{\mathrm{OG}}\boldsymbol{h}_{t-1} + \boldsymbol{p}_{\mathrm{OG}} \odot \boldsymbol{c}_t + \boldsymbol{b}_{\mathrm{OG}}), && \text{(A.9)}\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), && \text{(A.10)}
\end{aligned}$$

where $\boldsymbol{x}_t$ and $\boldsymbol{h}_t$ are external input and output from the layer at time step t, respectively. $\boldsymbol{a}_t$ is the input to the layer calculated by external input $\boldsymbol{x}_t$ and recurrent input $\boldsymbol{h}_{t-1}$. $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, and $\boldsymbol{o}_t$ are the input gate, forget gate, and output gate, respectively. The input gate controls how much $\boldsymbol{a}_t$ is fed into the memory-cell. The forget gate determines how much of the previous state of memory-cell $\boldsymbol{c}_{t-1}$ should be maintained. Finally, the output gate controls how the updated cell state can flow as the output of the layer. σ is the sigmoid function. Here, as long as the forget gate keeps the value close to 1, the memory cell state is not attenuated. Therefore, LSTM units can deal with longer temporal patterns than normal recurrent units. The propagation process is fully differentiable, so all the learnable parameters, i.e., external input connections $\boldsymbol{W}$, recurrent connections $\boldsymbol{R}$, peep-hole connections $\boldsymbol{p}$, and biases $\boldsymbol{b}$, can be optimized by the gradient descent algorithm.

Appendix B

Learning Details of Bell Task in Section 5.6

This appendix describes the details of the learning experiment that we performed in Section 5.6.

B.1 Data Representation

The task execution was represented as a series of 26 dimensional vectors. Table B.1 describes what each element represents. We assigned ten elements for words. Each sentence was represented as a series of one-hot vectors. To represent the robot arm joints, we also assigned ten elements. We recorded behavioral sequences by running programs that controlled the robot arm joints along predetermined trajectories. They take approximately 25 time steps in the case of SLOW behavior and 16 time steps in the case of FAST behavior. We assigned six elements to encode visual information. For instance, the pair of v_{l0} and v_{l1} represented the color of the left bell. In the current task, we assumed that the red, green, and blue bells corresponded to 0, $2\pi/3$, and $4\pi/3$ rad on the hue circle, respectively. The element v_{l0} took the cosine of the hue angle of the left bell, and v_{l1} was its sine. Similarly, the pairs v_{c0}, v_{c1} and v_{r0}, v_{r1} encoded the color of the center and right bells, respectively. We concatenated instructions and behavioral responses on a computer in a similar manner to the experiment in Chapter 4.

Table B.1. Data representation of the bell task.

Elements	Descriptions
w_0	"point"
w_1	"hit"
w_2	"red"
w_3	"green"
w_4	"blue"
w_5	"slowly"
w_6	"fast"
w_7	"and"
w_8	"or"
w_9	"not"
v_{l0}	cosine of the hue of the left bell
v_{l1}	sine of the hue of the left bell
v_{c0}	cosine of the hue of the center bell
v_{c1}	sine of the hue of the center bell
v_{r0}	cosine of the hue of the right bell
v_{r1}	sine of the hue of the right bell
j_{l0}	the shoulder pitch of the left arm
j_{l1}	the shoulder roll of the left arm
j_{l2}	the elbow roll of the left arm
j_{l3}	the elbow yaw of the left arm
j_{l4}	the wrist yaw of the left arm
j_{r0}	the shoulder pitch of the right arm
j_{r1}	the shoulder roll of the right arm
j_{r2}	the elbow roll of the right arm
j_{r3}	the elbow yaw of the right arm
j_{r4}	the wrist yaw of the right arm

B.2 Learning Setting and Evaluation Method

To train models, we made a dataset that consisted of 512 sequences. Each sequence included eight successive episodes. The dataset included all the possible episodes, which were defined as a combination of the bell arrangement, instruction, and behavioral response, at least once. We trained each of the single-context-layer models with 100, 300, 500, and 700 LSTM nodes ten times with different random seeds. The optimizer was Adam, whose learning rate was 0.001. The number of

learning iterations was 10,000.

We made another dataset to evaluate the model performance. It included all possible situations ten times. When the RMSE between the outputs and the ground truth per joint per time step during the behavior execution was less than 0.04, we judged that the model had successfully generated appropriate behavior. We regarded the situation patterns in which the model successfully generated appropriate behavior seven trials or more out of ten as "successfully learned."

B.3 Task Performance

We classified the episodes into four types: episodes in which an instruction with a single object word was given (72 situations); an AND-concatenated instruction was given (144 situations); an OR-concatenated instruction was given (144 situations); and a NOT-prefixed instruction was given (72 situations). We evaluated the models' performances by counting the number of situational patterns that the models successfully learned with respect to each of four types. Figure B.1 summarizes the resulting performance. An increase in the number of LSTM nodes improved the performance, but the 500- and 700-node models gave no significant difference in performance.

We also investigated which behavior the model chose for instructions that accept multiple behavior patterns as correct. The situations that have a plural number of correct behaviors are classified into three types: (a) When an instruction directs the model to act to the center bell, the model can act with either arm. Therefore, two different behaviors are accepted as correct. (b) When an instruction directs the model to act to the "left or right" bell, two answers exist. (c) When an instruction directs the model to act to the "left or center" or to the "right or center," three solutions exist: acting (1) to the left (right) bell with the left (right) arm, (2) to the center bell with the left arm, and (3) to the center bell with the right arm. Table B.2 summarizes the results for these three situational types. Here, we counted the result of the 500-node model. The model chose each of the different solutions evenly. Situation types (a), (b), and (c) include 24, 48, and 96 variations of situations, respectively, and we performed ten trials for each

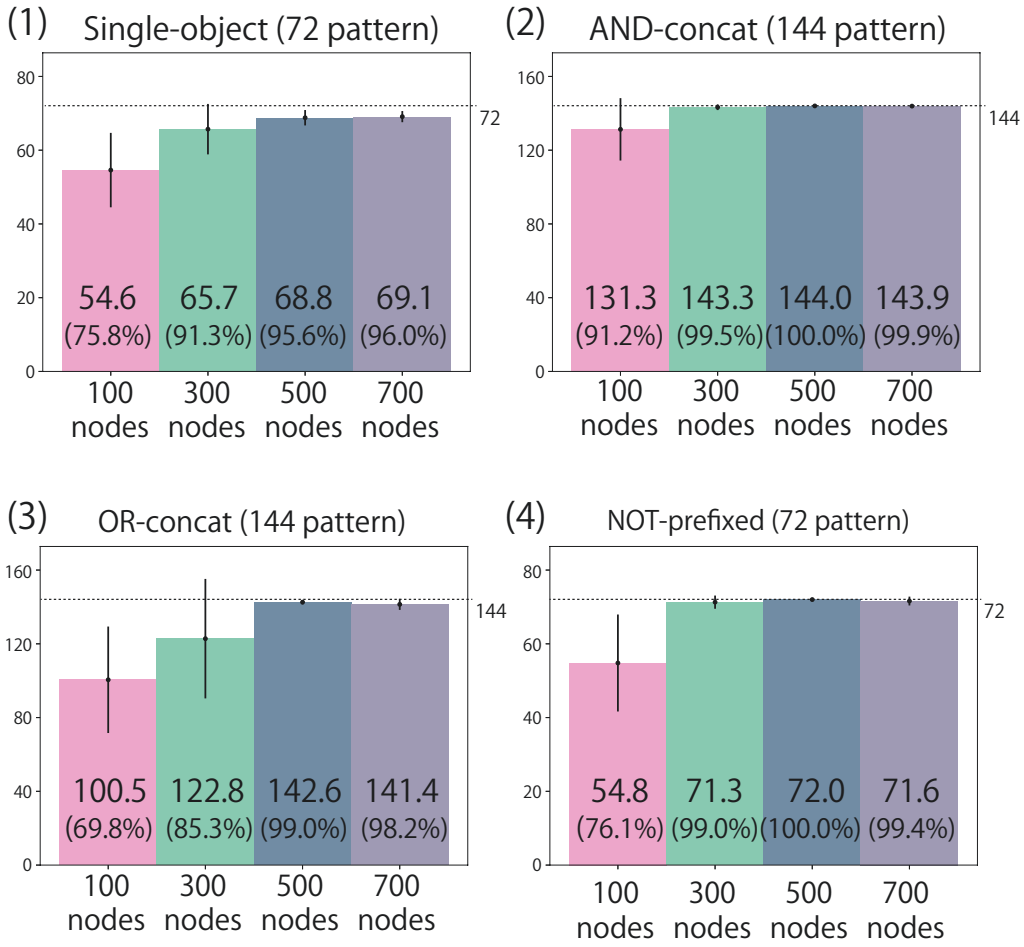

Figure B.1. Behavior execution performance. The performance was scored as the number of situation patterns the models successfully learned with respect to each of the four episode types: single-object instruction; AND-concat instruction; OR-concat instruction; and NOT-prefixed instruction. The values on the bars are the means of ten trials in which training was performed with different random seeds. Error bars visualize the standard deviations.

of these situations. In most of these ambiguous situations, the model chose all of the possible solutions at least once. In other words, the model successfully learned to behave appropriately in ambiguous situations, as in the flag task.

Table B.2. Ratios of chosen solutions for ambiguous instructions, which accept a plural number of solutions. The details of each situation type are described in the body text.

Situation	Choice 1(%)	Choice 2 (%)	Choice 3	Failure (%)
(a)	49.2	42.9	-	7.9
(b)	50.0	44.4	-	5.6
(c)	29.1	29.4	36.6	5.0

Relevant Publications

Journal Papers

1. Tatsuro Yamada, Hiroyuki Matsunaga, and Tetsuya Ogata. "Paired Recurrent Autoencoders for Bidirectional Translation between Robot Actions and Linguistic Descriptions," IEEE Robotics and Automation Letters, Vol. 3, Issue 4, pp. 3441-3448, 2018.

2. Tatsuro Yamada, Shingo Murata, Hiroaki Arie, and Tetsuya Ogata. "Representation Learning of Logic Words by an RNN: From Word Sequences to Robot Actions," Frontiers in Neurorobotics, Vol. 11, Article 70, pp. 1-18, 2017.

3. Tatsuro Yamada, Shingo Murata, Hiroaki Arie, and Tetsuya Ogata. "Dynamical Integration of Language and Behavior in a Recurrent Neural Network for Human–Robot Interaction," Frontiers in Neurorobotics, Vol. 10, Article 5, pp. 1-17, 2016.

International Conferences

1. Tatsuro Yamada, Hiroyuki Matsunaga, and Tetsuya Ogata, "Paired Recurrent Autoencoders for Bidirectional Translation between Robot Actions and Linguistic Descriptions," Proceedings of 2018 IEEE/RAS International Conference on Intelligent Robots and Systems, Madrid, Spain, October 2018.

2. Tatsuro Yamada, Shingo Murata, Hiroaki Arie, and Tetsuya Ogata, "Dynamical Linking of Positive and Negative Sentences to Goal-oriented Robot

Behavior by Hierarchical RNN,” In Artificial Neural Networks and Machine Learning — ICANN 2016 (Proceedings of the 25th International Conference on Artificial Neural Networks), Lecture Notes in Computer Science (LNCS), Alessandro E.P. Villa et al. (Eds.), Vol. 9886, pp. 339-346, Barcelona, Spain, September 2016.

3. 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, pp. 4179-4184, Hamburg, Germany, September 2015.

Domestic Conferences

1. 山田竜郎，松永寛之，尾形哲也，“共有表現の学習によるロボット動作と指示説明文の双方向変換，”第 32 回人工知能学会全国大会，鹿児島，2018 年 6 月．

2. 山田竜郎，村田真悟，有江浩明，尾形哲也，“Seq2seq 学習による論理語を含む言語指示の理解とロボット行動の生成，”第 31 回人工知能学会全国大会，愛知，2017 年 5 月．

3. 山田竜郎，村田真悟，有江浩明，尾形哲也，“階層型 RNN を用いたロボットの旗揚げタスクにおける肯定及び否定指示の理解，”第 30 回人工知能学会全国大会，福岡，2016 年 6 月．

4. 山田竜郎，村田真悟，有江浩明，尾形哲也，“ターンテイキングタスクを行うロボットのための神経回路力学系上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，1B3-05，東京，2015 年 9 月．

5. 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．

Other Publications

International Conferences

1. Tatsuro Yamada, Tetsuro Kitahara, Hiroaki Arie, and Tetsuya Ogata, "Four-part Harmonization: Comparison of a Bayesian Network and a Recurrent Neural Network," Proceedings of the 13th International Symposium on Computer Music Multidisciplinary Research (CMMR2017), pp. 137-148, Porto, Portugal, September 2017.

2. Tatsuro Yamada, Saki Ito, Hiroaki Arie, and Tetsuya Ogata, "Learning of Labeling Room Space for Mobile Robots Based on Visual Motor Experience," In Artificial Neural Networks and Machine Learning — ICANN 2017 (Proceedings of the 26th International Conference on Artificial Neural Networks), Lecture Notes in Computer Science (LNCS), Alessandra Lintas et al. (Eds.), Vol. 10613, pp. 35-42, Alghero, Italy, September 2017.

3. 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.

Domestic Conferences

1. 澤弘樹，山田竜郎，村田真悟，森裕紀，尾形哲也，菅野重樹,“RNN を備えた二台ロボット間インタラクションの複雑性解析，”情報処理学会 第 80 回全国大会，東京，2018 年 3 月．

2. 張耀宇，中條亨一，山田竜郎，村田真悟，有江浩明，尾形哲也,“ 神経回路モデルにおける追加学習手法に関する検討，”第 18 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，宮城，2017 年 12 月．

3. 伊藤彩貴，山田竜郎，有江浩明，尾形哲也,“ 深層学習を用いた移動ロボットによる室内空間の状況依存的ラベリング，”第 35 回日本ロボット学会 学術講演会，埼玉，2017 年 9 月．

4. 山田竜郎，北原鉄朗，有江浩明，尾形哲也,“ LSTM を用いた四声体和声の生成，”第 31 回人工知能学会全国大会，愛知，2017 年 5 月．

5. 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．