Doctoral Dissertation (Censored)

博士論文（要約）

Disentangling animal behavior with probabilistic generative models

(確率的生成モデルを用いた動物行動の解析)

A Dissertation Submitted for the Degree of Doctor of Philosophy

February 2021

令和 3 年 2 月博士（理学）申請

Department of Biological Sciences, Graduate School of Science,

The University of Tokyo

東京大学理学系研究科

生物科学専攻

Keita Mori

森　啓太

Abstract

Animals exhibit a variety of behaviors that are adapted to their environment. Behavior is the final output of the nervous system and is a major factor that determines the survival of an individual. Understanding the generation and control processes of such diverse and important behaviors is a major goal not only for ethology, but also for neuroscience, ecology, evolution, and informatics, and it is a problem that has not yet been solved. The main goal of this research is to elucidate and reproduce the system that can generate and control various behaviors seen in nature.

Acquisition of the behavior of freely moving *Caenorhabditis elegans*

In this dissertation, the behavioral data were mainly quantified and analyzed using the nematode *Caenorhabditis elegans* (*C. elegans*) to confirm the validity of the method. *C. elegans* is a suitable experimental animal to test the effectiveness of this study because it shows stochastic and multiple behavioral patterns while the behavior is easy to measure. First, a behavioral data set was obtained to test the developed method, and the behavioral state was quantified by recording *C. elegans* freely moving in a two-dimensional plane. In addition, to investigate whether the stochastic behavioral responses to sensory stimuli can be appropriately modeled, I expressed channelrhodopsin (ChR2) in ASH neurons, which are sensory neurons to nociceptive stimuli, and activated them by randomly exposing them to blue light during behavioral recordings. From the acquired images, I quantified posture and speed as indicators of behavior using image processing.

Probabilistic generative neural networks disentangle dynamics of animal behavior

In Chapter 1, I aimed to develop a virtual animal model which can both reproduce stochastic animal behavior and represent various behaviors in a disentangled latent space. In order to model a phenomenon that takes multiple states and is stochastic, a mixture density network - recurrent neural network (MDN-RNN), is employed. The MDN-RNN is a simulation-based modeling method to model time series data with stochastic

behavior by using a probability distribution as the output part of the RNN. Using MDN-RNN, I trained a simulator to predict the behavior of *C. elegans* 0.2 seconds later based on the behavior of the past 20 seconds. After training the simulator using MDN-RNN, it was confirmed that the simulator generated behaviors with the same behavioral states and dynamics as those of the real *C. elegans* through analysis using t-SNE and time-delay embedding methods.

In order to understand the behavior that consists of multiple stochastic states, I analyzed the internal representation of the model that can generate the behavior similar to that of a real *C. elegans*. The results showed that each MDN component represented different behaviors. As a result, it was found that the dynamics of different behavioral states were extracted and represented in each component.

Behavior control via reinforcement learning

In Chapter 2, I developed a computational control mechanism for behavior, and showed that a machine can extract behavioral strategies found in nature by automatically searching for a control mechanism that suits the desired task. In order to achieve this, I applied reinforcement learning as a control algorithm, and aimed to replace some of the computational mechanisms of the nervous system with a computer in order to accomplish the tasks that animals perform in nature. It was shown that the computer can acquire and reproduce behavioral strategies similar to those actually performed by animals without prior information.

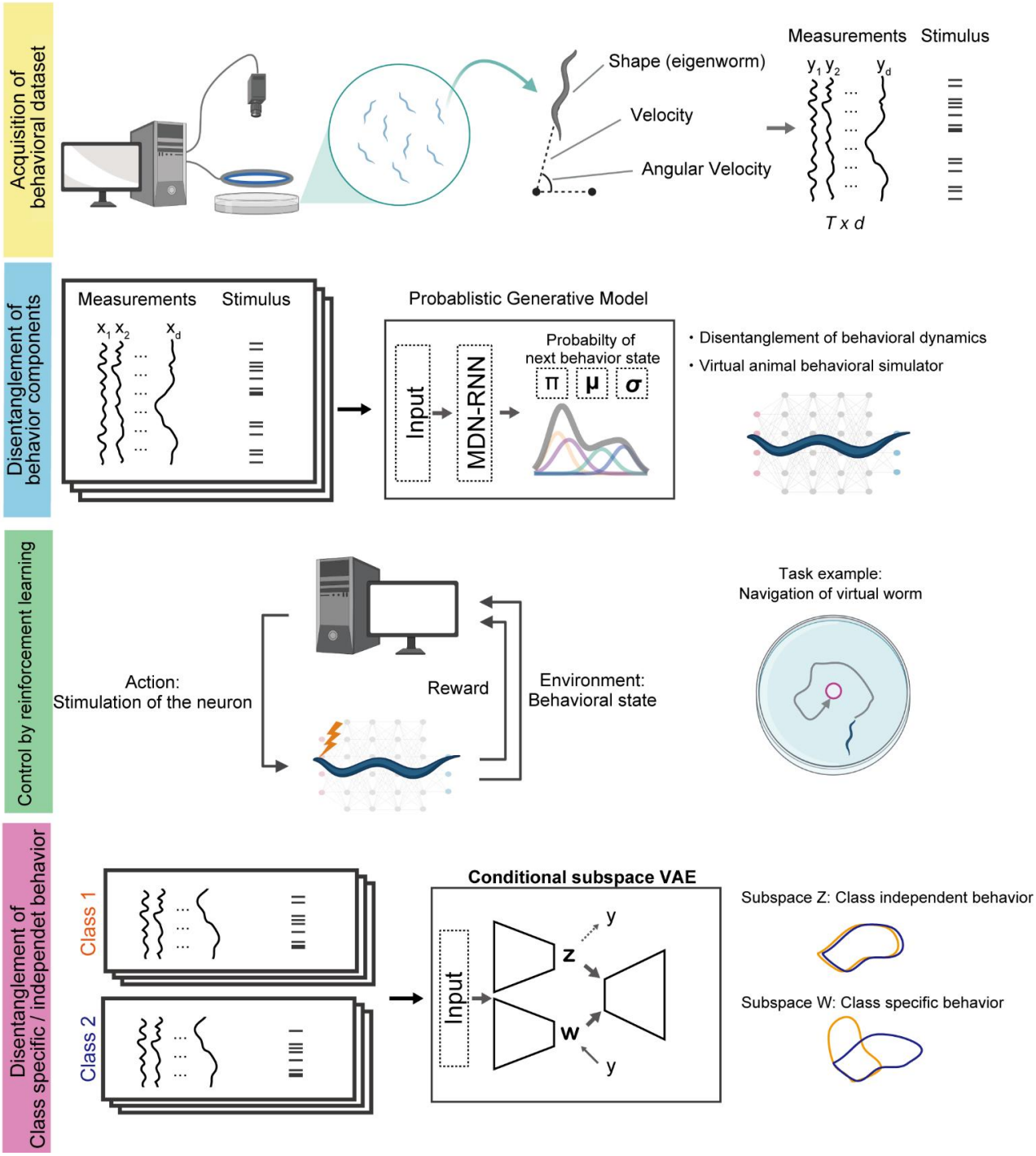
Disentangling animal behavior via temporal conditional-subspace VAE

In Appendix A, I aimed to analyze the topology of the dynamics behind the behaviors of animals belonging to different classes by separating the behaviors that are characteristic of each class from the behaviors that are common regardless of the class. For example, when the behaviors of wild-type animals and model animals for psychiatric disorders are acquired, it is important to extract the behaviors exhibited only by the disease model animals and investigate the generation mechanism of the behaviors in order to clarify the

disease. To achieve this goal, I applied the conditional subspace - variational autoencoder (CS-VAE), which takes behavioral data consisting of multiple groups as input and divides them into elements characteristic of the group to which they belong and elements common to all groups in the middle layer of the VAE. The CS - VAE takes behavioral data consisting of multiple groups as input and divides it into elements characteristic of the group to which it belongs and elements common to all groups in the middle layer of the VAE. This is achieved by minimizing the amount of mutual information between the labels of the groups to which they belong and the features of the latent space in the middle layer. The effectiveness of this method has been verified using toy models, and will be verified using animal behavior in the future.

Conclusion

In this dissertation, I have succeeded in separating animal behaviors with probabilistic and multiple control states by using representation learning of deep generative models. I also succeeded in automatically learning a policy to control a virtual animal by reinforcement learning. In this study, *C. elegans* was used as a model animal, but this model can be applied to other animal species as well. I aim to further develop these methods to elucidate the process of generating behavior from neural activity in an interpretable manner.


Acquisition of behavioral dataset
Shape (eigenworm)
Velocity
Angular Velocity
Measurements
Stimulus
T x d
Disentanglement of behavior components
Measurements
Stimulus
Probablistic Generative Model
Input
MDN-RNN
Probabilty of next behavior state
• Disentanglement of behavioral dynamics
• Virtual animal behavioral simulator
Control by reinforcement learning
Action:
Stimulation of the neuron
Reward
Environment:
Behavioral state
Task example:
Navigation of virtual worm
Disentanglement of Class specific / independet behavior
Class 1
Class 2
Conditional subspace VAE
Input
Subspace Z: Class independent behavior
Subspace W: Class specific behavior

Contents

General Introduction

In the 4.4 billion years since the birth of life and the 3.8 billion years since the emergence of diverse organisms in the Cambrian Explosion, animals have acquired diverse and complex behaviors in the course of evolution. Animal behavior is the ultimate output of the nervous system, serving as the interface with the surrounding environment. As animals survive by interacting with their surrounding environment, proper behavioral control is critical for survival. It is a major goal of behavioral neuroscience (and other related fields) to clarify the mechanisms that generate these various behaviors, as well as the control mechanisms. Conceptual guidelines for studying animal behavior were proposed by Tinbergen in the middle of the 20th century; these guidelines have since evolved, along with the surrounding fields, including neuroscience, psychiatry, psychology, information science, and robotics. In nature, there is a great variety of behaviors, with each behavior influenced by multiple factors, such as genetics, previous experiences, and surrounding environments. Why these behaviors occur, what mechanisms control them, and how they have been acquired during evolution are fascinating questions. Furthermore, the purpose, control algorithms, and implementation methods (Marr's three levels [1]) of these behaviors have received great attention in the fields of artificial intelligence and robotics. However, it is difficult to describe each behavior that arises from the diverse and complex factors found in nature, as well as to search for the factors that may affect it. These difficulties call for a systematic method of analysis.

Until now, the main strategy has been to observe animal behavior and describe characteristic behaviors, and then search for factors that may influence behavior and investigate the relationship between the factors and the output in a hypothesis-driven manner. This approach is still important, but it is costly when dealing with diverse systems, and is affected by observer bias. However, with the developments in technology, especially measurement technology, achieved over the past few years, we are now ready to analyze animal behavior systematically. In the following sections, I will review the important developments in measurement technology, and the computational analysis methods that have emerged as a result.

The rise of computational ethology

Improvement of quantification methods

The field of ethology has recently entered a new phase due to progress in measurement technology. There have been two major technological breakthroughs. The first breakthrough comprises progress in measurement hardware; the recent development of low-cost, high-resolution cameras and widespread use of three-dimensional (3D) printers have facilitated the creation of behavior measurement devices. The second breakthrough comprises advancements in machine learning and image processing technologies, which have facilitated the quantification of posture and other features. I will not detail these developments; rather, I will briefly describe them and provide references for further reading.

Developments of hardware

The democratization of hardware development in recent years has had a profound impact on the field of ethology. The widespread availability of low-cost 3D printers and microcomputers (e.g., Arduino, Raspberry Pi, and Jetsons) has rendered it easy for laboratories to develop their own devices, tailored to the behaviors they wish to measure. Furthermore, high-performance cameras can be obtained at low cost; for example, cameras with depth sensors (e.g., Realsense) are available worldwide. In addition, field programmable gate arrays and graphics processing units for real-time image processing are becoming popular, and experimental systems such as the Etholoop (Fig. 1)[2], which tracks animal behavior in 3D in real time and stimulates the animal's nervous system in a closed-loop manner, may also become popular. Furthermore, the sharing of hardware, software, and knowledge in an open-source manner is important to the continued development of this field and community.

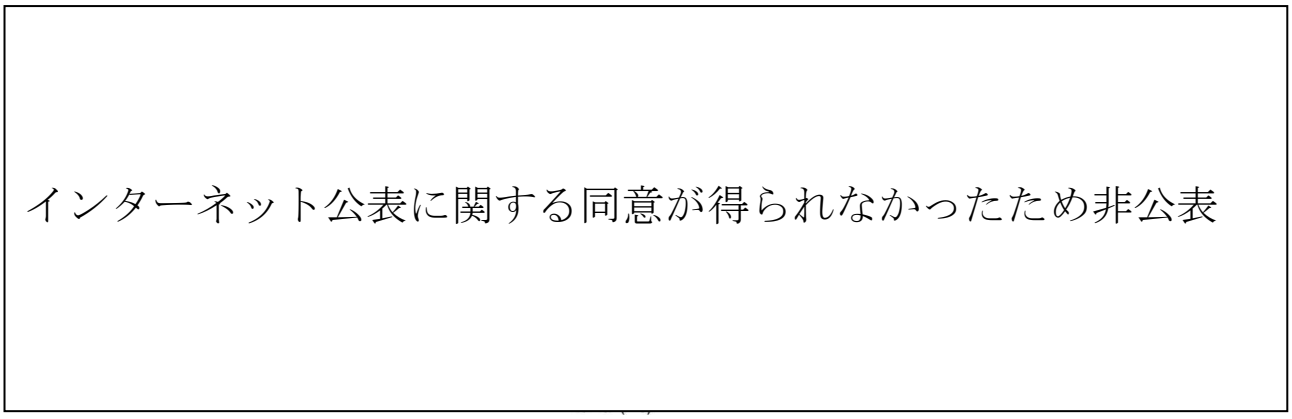


Fig. 1 Etholoop system[2].

Developments of software

Additionally, recent dramatic improvements in markerless tracking of body parts have enabled a range of exciting new possibilities for potential studies (Fig. 2). A typical example of a deep learning model for markerless tracking is deeplabcut[3]. See these articles and reviews[3–5] for details.

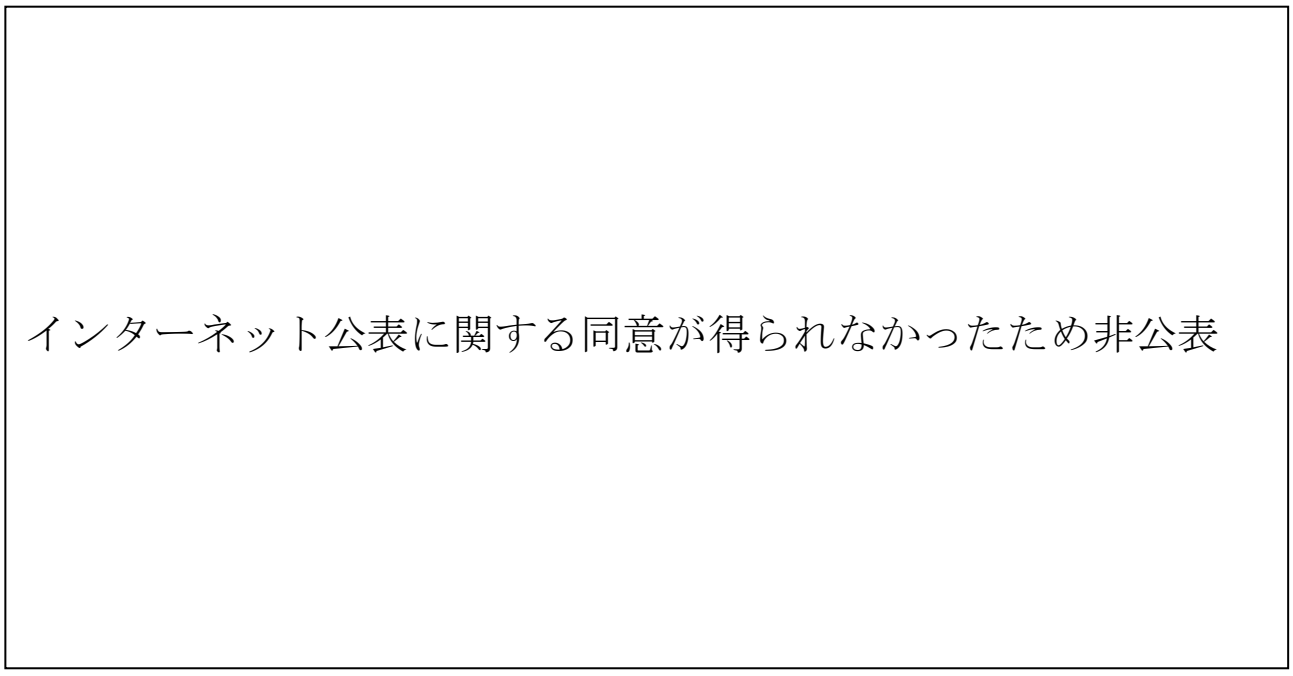


Fig. 2 Development of motion tracking softwares[4].

Although the modern recording and posture estimation techniques mentioned above can generate large-scale measurements of multiple animal behaviors in both laboratory and naturalistic environments, elucidating the underlying processes that comprise behavior remains challenging. This process corresponds to the first and second steps in Marr's framework (computation and algorithm) [6]. Furthermore, answering each of Tinbergen's 4 questions [7]) is an important step toward the understanding of animal behavior. Several threads of computational works have tackled this problem from different points of view. In the following sections, I will introduce analytic methods useful for analyzing the obtained datasets. Additionally, in the final section, I will describe the current methodological challenges and possible future directions of the field.

Automatic supervised behavior labeling

Ethologists and behavioral neuroscientists have described and statistically analyzed behavioral patterns by focusing on behavior motifs (also known as behavioral "syllables") or behavioral “grammars” (which are the transition patterns of behavioral syllables), just as an unexplained language is broken into its grammars [8–10]. For decades, the counting and statistical analysis of these motifs has been commonly performed not only in behavioral neuroscience, but also in human science and research. However, previous studies have manually annotated the behavioral motifs and it was time consuming.

The simplest example of automation by machine learning is the automation of behavioral motifs labeling. This work has been conducted in both ethology and human science [11–19]. Several computational approaches have been used for supervised clustering, including the decision-tree method (or random forest ensembles), Gaussian mixture models, and neural networks-based methods.

Unsupervised behavioral motif classification

The automatic classification of behavioral patterns can also be achieved using unsupervised classification, i.e., the experimenter provides the machine with only behavioral data, and the machine

automatically learns plausible classification patterns. Traditionally, ethology and behavioral neuroscience have relied on the summary statistics of handcrafted criteria. However, even in highly controlled conditions, these metrics tend to be unreliable (across animals, laboratories, and experimenters) [20–22]. Furthermore, relying on the prior knowledge of a researcher (with inherent bias) may fail to capture complex relationships and patterns in higher-dimensional descriptions or higher-order temporal patterns. These limitations have prompted interest in developing data-driven methods. Instead of an observer finding behavioral patterns, machine learning calculates the similarities among the data of individuals and classifies them. This is expected to minimize observer bias and enable quantitative methods that are reproducible.

Behavior map (clustering)

A typical method for classifying animal behavioral patterns in an unsupervised manner from time-series data is the behavior map (Fig 3). Since it was proposed in 2014 to visualize the behavior of *D. Melanogaster* [23], the behavior map has been widely used in various animal species, such as *C. elegans*[24], zebrafish [25], and rodents. A standardized pipeline is typically used to create the behavior map. Spectral estimation using wavelet transformation [26] and manifold embedding techniques (such as t-stochastic neighbor embedding[27]) are applied to raw behavioral data, and embedded points are classified via Gaussian mixture models and k-means clustering [23]. In combination with probability density functions, it is also possible to quantify changes in the frequency of behavioral patterns using concepts of information content, such as entropy [28].

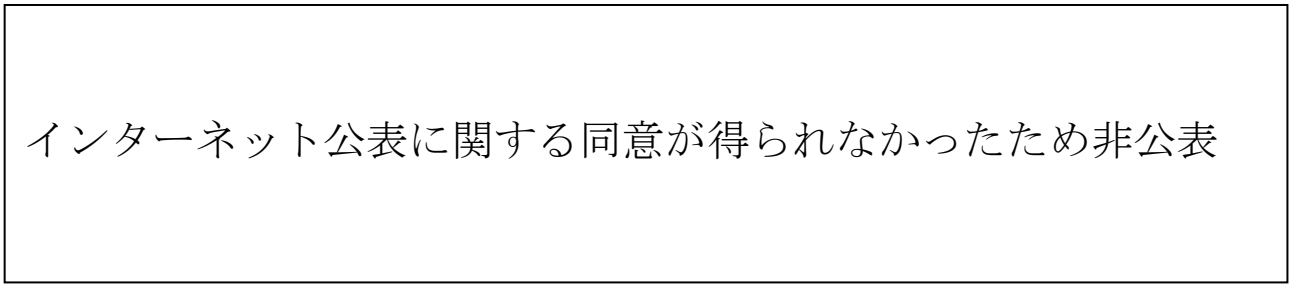


Fig. 3 Schematic illustration of behavior map creation[23].

Interpretation of behavior

The next step in behavior motif quantification, towards an understanding of the behavior, is to describe the structure of the behavior dynamics and infer the underlying control mechanisms. In behavior classification, there is little interpretation of the structure underlying behavior dynamics. To assess the dynamics or control mechanisms behind the observed behavior, a more sophisticated approach is required to represent the behavior dynamics.

In this section, I first briefly introduce the major approaches in dynamics analysis, including hidden Markov model (HMM)-based, physical science-based, and deep generative model-based approaches, followed by a detailed discussion. The HMM-based approach assumes that there is a hidden state underlying the behavioral raw data, and aims to understand the underlying structure by inferring the transition patterns of the hidden state. In the physical science-based approach, behavior is considered as a dynamical system, and the underlying structure is inferred by using dynamical system analytic methods. The deep generative model-based approach aims to build a model that can generate data similar to the obtained behavioral data, and the underlying structure is inferred by the generative process of the model, or by disentangling the representations inside the model. It is important to note that although these methods have different starting points, they are not antithetical to each other and can be used in combination. Therefore, it is important to understand the nature of each method and use or combine them according to their purpose.

Hidden Markov model (HMM) based method

State-space models using a HMM have long been utilized in the analysis of time-series data. HMM-based methods generally model the observed behavioral data by assuming discrete hidden 'states' that parametrize the generation process underlying the data. HMMs are highly interpretable because each phenomenon is modeled by a discrete state with different parameters governing the behavioral dynamics. HMMs also have a long history in modeling human behavior [29,30]. A general explanation of HMMs is provided in the relevant reference [31], and specific applications to ethology are introduced below.

One HMM application is known as Motion Sequencing (MoSeq) [32], which can break behavioral sequences into a set of reused and stereotyped sub-second behavioral motifs. MoSeq combines 3D imaging techniques with an autoregressive-HMM and characterizes the nature of behavioral changes among behavior motifs. Furthermore, MoSeq effectively parses behavioral differences and captures similarities elicited by the pharmacological treatments or genetic factors [33].

Another area of research in HMM-based computational modeling extends HMM to handle a continuous latent space. It is difficult to represent continuous time-series data, such as posture transitions and neural activities, with a HMM alone. This limitation prompted the development of the switching linear dynamical system (SLDS), which combines a HMM and linear dynamical system (LDS) [34–40], allowing discrete switches to depend on a continuous latent space and external input. The SLDS is a model in which each discrete state has different LDS parameters (corresponding to the dynamics), and the behavior of the system changes as the discrete state transitions according to a Markov transition matrix. Linderman and colleagues [41,42] developed an extension of the SLDS, known as a recurrent SLDS (rSLDS). In an rSLDS, the transition probability is parameterized by the position in continuous space, leading to a more natural transition. Additionally, models that can handle neural activity and/or animal behavior have been developed [43–45]. Through a series of these studies, Linderman's group has applied these methods to the analysis of behavior and neural activity in *C. elegans*, mice, primates, and other animals.

Ethology from a physical science standpoint

From a physical science standpoint, animal behavior can be viewed as a time-evolving dynamical system in a space with a high degree of freedom, including complex posture dynamics. In this high-dimensional space, animal behaviors are thought to move along a trajectory within a specific attractor due to certain constraints, and determining the type of attractor is the very essence of investigating the dynamics behind animal behaviors.

One of the strongest tools used in attractor analysis is time-delay embedding, supported by the "embedding theorem" presented by Takens [46]. This theorem serves as a bridge between the theory of dynamical

systems and actual measured time-series data. Time-delay embedding enables the reconstruction of the topological features underlying time-series data; by monitoring the geometrical features, the dynamics of the system can be characterized [47]. Ahamed and colleagues [47] developed a reconstruction method that considers animal behavior as a time-evolving dynamical system, and combined it with an independent component analysis (ICA) to analyze the mechanisms underlying animal behavior. In this method, the obtained multivariate time-series data is reconstructed with time-delay embedding, and is then divided into behavioral elements with different dynamics (differential equations) based on the ICA results. Using this method of embedding and ICA, the obtained multivariate measurements can be smoothly unfolded as a combination of short-time posture sequences. Furthermore, Tran and Hasegawa [48] showed that the combination of a topological analysis and delay-variant embedding, which considers the time delay as a variable parameter, can successfully classify the behavior of *C. elegans*. These studies demonstrate that delay embedding is a powerful tool that can characterize dynamical systems and provide a topological analysis of the trajectories in embedded space, enabling insights into the geometric structure underlying behavior.

Representation learning via Deep generative model

One of the most interesting advances in the machine learning field in the 2010s arose through novel applications of deep learning to generative modeling tasks. Generative modeling attempts to learn the underlying structure of the data generation process. The ability to learn the generative mechanisms behind the data renders this method a good fit for science, and it is beginning to be used in the fields of animal behavior and neuroscience, to infer the structure behind observed events [49]. The strength of deep generative models is that the model learns disentangled meaningful representations of observed data during the training process (i.e., representation learning). Typical models used in representation learning comprise encoder-decoder-based models, such as the variational autoencoder (VAE). VAE and its variants are mainly used in science because of their high interpretability [35] and extendibility of the model after learning. Several studies have been conducted in the context of human behavior prediction using motion capture (mocap) data [50] and in the ethology field, as discussed below.

Human motion prediction from mocap data is a classical problem in the field of computer vision. Learning meaningful representations of human motion plays an important role in many practical tasks, such as human behavior prediction, human-computer (or robot) interaction, and the production of games and movies [49,51–53]. Recently, it has been shown that an autoencoder-type neural network can learn the manifold of human motion [54], and can represent different motions in a latent space, in a disentangled form[55] . These approaches utilize the representation learning feature of autoencoders, and succeed in obtaining the manifold underlying the behavioral dynamics.

In ethology, there are several reports that utilized deep generative models for understanding the underlying dynamics of animal behavior, including the VAE-stochastic neighbor embedding[56] and BehaveNet[57]

Representation learning using deep generative models is a very powerful tool and further developments are expected. The analysis of human mocap data and that of animal posture data are essentially similar, and it is important to share knowledge between these fields. In addition, the method of separating and modeling multiple generative processes is important for both neurobehavioral science and the imitation of behavioral control mechanisms in robotics.

Future direction

Linking brain activity and behavior

To fully elucidate the mechanisms that generate and control animal behavior, we need to understand how the algorithms underlying the behavior are computed and implemented by the nervous system. This is the goal of many scientists in behavioral neuroscience. There has been progress in the field of neuroimaging, and it is now possible to simultaneously acquire behavioral and neuronal activity data from model animals that are freely moving. Although we have already begun to map the dynamics of numerous neural activities to the dynamics of behavior, further developments in computational methods are required to elucidate this link and to obtain a meaningful representation of the control mechanism.

Computational method for comparative research

Comparison is a very powerful method in enabling humans to understand objects, and behavioral comparison is an important subject of analysis in ethology and related fields. In the field of genetics, behavioral changes caused by genetic mutations are compared, and in the study of psychiatric disorders, it is important to accurately describe behavioral changes in the presence and absence of disease, as well as those caused by pharmacological treatments. In developmental biology, it is important to understand how behavior changes during development, and from an evolutionary perspective, which behaviors have changed among closely related species. Whereas researchers previously focused only on indices handcrafted by the observer, methods have now been developed to express the dynamics underlying behavior in an unsupervised manner with high interpretability. Thus, we will be able to handle essential differences that have not been revealed as yet. For this purpose, it is important to have a method to measure the differences (distances) among obtained expressions. This may be realized by measuring the distance between probability distributions, or by using adversarial learning.

Considering probabilistic feature of behavior

It is not easy to model natural events, including animal behavior, as they are stochastic from the observer's point of view, and although HMMs and VAEs can handle stochastic elements at a high level of underlying abstraction, stochastic elements occurring downstream are often not considered. For example, in the field of human motion prediction, recurrent neural networks are often used, but they cannot fully represent probabilistic elements and tend to degrade into motionless states or drift away to non-human like motions. In order to solve this problem, I propose the use of probability distributions as the prediction target, or to treat the parameters of each information-processing process as probability distributions and perform Bayesian estimation in the framework of graphical modeling.

Control of behavior and closed loop experiments

Beyond the analysis of animal behavior, it is possible to develop methods to control behavior. It is already possible to intervene in nervous system activity through optogenetics and electrical stimulation. In the future development of the brain-machine interface, the importance of methods such as augmentation, in which the computer and nervous system interact to control behavior and improve the processing power of the nervous system, will increase. After understanding the original control mechanisms in animals, an upper layer comprising computer control mechanisms will need to be added. It will be important to develop a method to seamlessly connect the meaningful representations accumulated in computational ethology with these control mechanisms.

Aim of this study

Three key problems in computational ethology were approached in this study.

In Chapter 1, a simulation method for reproducing stochastic behavior of animals has been developed. The construction of a virtual animal model that reproduces the mechanism of animal behavior is important for understanding animal behavior. In particular, since the connectivity of each neuron is known in *C. elegans*, it is expected that a virtual animal model will be constructed before other model animals. In this study, I developed a method to reproduce stochastic behavior by combining a deep generative model and a generative model that predicts probability distributions. By using representation learning, which is an advantage of generative models, it was shown that different behavioral patterns (i.e., behavioral patterns with different dynamics) are modeled by different components in the model.

In Chapter 2, I developed a computational control mechanism for behavior, and showed that a machine can extract behavioral strategies found in nature by automatically searching for a control mechanism that suits the desired task. In order to achieve this, I applied reinforcement learning as a control algorithm, and aimed to replace some of the computational mechanisms of the nervous system with a computer in order to accomplish the tasks that animals perform in nature. It was shown that the computer can acquire and reproduce behavioral strategies similar to those actually performed by animals without prior information.

In appendix A, the development of a computational method for comparative behavioral studies, which will become increasingly important in the field of behavioral studies, is proposed. By incorporating the concept of adversarial learning, a method for separating behaviors with common dynamics from different behaviors in a comparison group is proposed and tested using a toy model.

Chapter 1

Probabilistic generative neural networks disentangle dynamics of animal behavior

1.1 Abbreviation

C. elegans	*Caenorharbitis elegans*
HMM	Hidden Markov model
ICA	Independent component analysis
LSTM	Long short term memory
MDN	Mixture density network
MDN-RNN	Mixture density network - recurrent neural network
NGM	Nematode growth medium
NN	Neural network
PCA	Principal component analysis
RNN	Recurrent neural network
t-SNE	t-distributed Stochastic Neighbor Embedding

1.2 Introduction

Understanding animals' behavior is of great importance in various fields including but not limited to ethology, neuroscience, and robotics [58–60]. From a biological standpoint, replicating animal behavior is important for understanding how the nervous system controls an animal's body and how it processes sensory information and makes decisions to generate corresponding motion. From an engineering standpoint, animal-inspired robots are in heavy demand for various applications [60–62]. In addition, understanding and predicting human behavior is important for building human-like robots and developing human assistance systems in real worlds. To that end, methods for modeling complex and stochastic behaviors are needed.

In general, the following steps are necessary to understand the control mechanism behind a system: observing the behavior of the system, constructing simulation models of the behavior, and then controlling the system *ad arbitrium* [63]. First, it is important to quantify the behavior of the system by observation. Next, the underlying control mechanism can be inferred by building a model that can reproduce the behavior of the system. Finally, understanding of the behavior is deepened if we learn how to control the system. Furthermore, by being able to control the system, we will be able to use the system in the real world. These series of methods are applicable to understanding the underlying mechanisms of animal behavior as well as other systems.

High-throughput behavioral measurement, which is the first step of computational study of behavior, has become pervasive across modern ethology and neuroscience thanks to the recent advancement of technologies including low-cost cameras, computer vision algorithms, and machine learning tracking techniques [3–5]. However, modeling behavior, the second step of computational ethology study, remains a challenging problem due to the complexity, diversity, and stochasticity of behavior, and there is substantial room for improvement in the field. Over the past few decades, researchers have proposed several methods for simulating animal behavior. Hidden Markov-based models and it's variation have been used to model the stochastic transition between several discrete behavioral patterns [41,43]. Additionally, the point process model has been used for modeling stochastic time evolution [64]. Neural network-based models such as recurrent neural networks have been used for modeling sequential and deterministic dynamics. However, simultaneous modeling of the continuous sequential features and stochastic features of behavior is a non-trivial task [65].

To overcome this difficulty of modeling stochastic behaviors, I propose an application of mixture density recurrent neural networks (MDN-RNNs). MDN-RNNs are recurrent neural networks (RNNs) combined with a mixture density network (MDN) which outputs parameters of a Gaussian mixture model [66,67]. MDN-RNNs process sequential inputs (past behavioral states) and output parameters of the Gaussian mixture model (the probability distribution of future behavioral states). MDN-RNNs have been used for prediction and generation of sequential data in several research areas such as future prediction of handwriting [67], sketch drawing [68], speech synthesis [69], and music generation [70] among others [71,72]. These studies showed that MDN-RNNs are suitable for simulating stochastic sequential data. I expect RNN to capture sequential characteristics and MDN to capture the stochastic characteristics of the behavior.

To investigate the effectiveness of MDN-RNNs as an animal behavior model, I used the nematode *Caenorhabditis elegans* (*C. elegans*), which is a widely-used model organism in the field of neuroscience and behavioral studies. *C. elegans* has ideal characteristics suitable for testing the new methods: their behavior can be easily recorded and described with only a few parameters, but are still quite complex and stochastic at the same time [73]. With their simple body shape and slow moving speed, automatic posture tracking is not difficult [74]. Several studies have investigated the behavioral states of *C. elegans* and have shown the stochasticity of their behavior [24].

1.3 Related research

I build this work upon several pieces of research on MDN and stochastic sequence prediction with MDN-RNNs. I will briefly introduce the concept of related research.

1.3.1 Mixture density network

MDNs were originally proposed by Bishop [66] for modeling a mixture of Gaussians with neural networks which were applied to solve the robot kinematics problem. MDN is an NN which uses probability distribution as its output (Fig. 1-2). It is suitable for prediction including confidence level or prediction of probabilistic events.

1.3.2 Modeling behavior with RNN

RNNs have been used in behavior prediction, especially in the task of human pose prediction [49,53,75,76]. However, for relatively long-term prediction, previous methods tend to fall into a motionless state or a state that cannot actually happen [49].

1.3.3 Modeling stochastic sequence with MDN-RNN

MDN-RNN is an NN that processes the output of RNN by MDN and outputs probability distribution (Fig. 1-3). while RNN predicts the state of the next time deterministically, MDN-RNN predicts it as mixture probability distribution (Figure). This allows us to deal with stochastic natural phenomena in a more original way.

MDN-RNNs were introduced to model handwriting [67] and sketch drawing [68]. Other applications include music generation [70], speech synthesis [69], and simulating a 2D game environment as "World Models"

[71]. Extending the "World Models" work, Ellefsen et al.[77] shed light on disentangled scene representation with MDN-RNNs. This work is inspired by these threads of works.

1.4 Materials and methods

1.4.1 Acquisition of the behavior of freely moving *C. elegans*

Multi-worm tracker assay

In this study, the behavior of *C. elegans* was used to verify the effectiveness of the method [78]. The behavior of freely moving adult hermaphrodites on the surface of an agarose plate was video recorded as follows. *C. elegans* hermaphrodites were cultured at 20 °C on NGM plates including 50 mM NaCl with *E. coli* as food source for 4 days and then the behavior was captured on assay plates with 50 mM NaCl for 30 min. I recorded the behavior of 20 to 30 worms simultaneously. The *ASH-ChR2* strain: *Is[Psra-6::ChR2::mCherry+Pges-1::EGFP]; lite-1(ce314) X* was used in this study. Random impulses of blue LED light were given to activate ASH neurons, and the images of the worms were captured simultaneously. Image capture speed was 5 frames per second (fps) and 9000 frames were captured in one assay. 1 pixel in the images is equal to 0.01 mm.

Behavioral quantification

Captured images were analyzed and the behavioral states (shown in Table) were calculated by using Matlab (R2017a) and Fiji [79]. The images were processed by a denoising median filter, background subtraction, and thresholding using automatically selected methods implemented in Fiji. Too small and too large objects were regarded as dusts on the assay plate and aggregations of multiple worms, respectively, and removed. The remaining objects were regarded as worms. The center lines of worms were obtained by skeletonizing using the bwmorph thinning function in Matlab. Coiled worms were detected and removed. The eigenworm component weights a1-a5 were obtained by projecting the center lines to the eigenworm space based on high-resolution tracking data. The worms were tracked by linking the nearest objects in the neighboring frames. The speed and direction of the worms were obtained from temporal differences of the centroids of the objects.

The behavior of *C. elegans* was characterized by assessing its posture, velocity, and angular velocity. The posture of worms can be described by the weighted sum of five principal components called eigenworms [73]. The posture was expressed in five dimensions using the eigenworm [73]. The velocity was expressed in three dimensions: speed of the center and direction of movement (sine and cosine). The angular velocity of the center was represented as a scalar. The behavior of *C. elegans* was quantified in a total of nine dimensions. For the convenience of tracking, data at times when posture and movement speed could not be calculated correctly were excluded, and data from worms in which behaviors could be quantified for less than 20 sec in a row were excluded.

1.4.2 Basic analysis of the behavior

Unsupervised behavioral classification by t-SNE

A nonlinear dimensionality reduction method, t-SNE, was used to visualize behavioral states. Based on the time series behavioral data consisting of seven dimensions of *C. elegans* posture, velocity and angular velocity, the data were pre-processed by PCA and then embedded in a two-dimensional space by t-SNE. After that, probability density estimation and Gaussian filter processing were performed to create a behavioral map. To add simulator-generated behavioral data to the behavioral map created from the real behavior we used a function of the openTSNE package [80].

Dynamics analysis by time-delay embedding

One of the methods to describe the behavior of a target system in a time-evolving nonlinear dynamical system is the time-delay embedding method based on Takens' theorem [46]. Other authors recently proposed a method to describe the dynamics of animal behavior by combining the time-delay embedding and dimensionality reduction methods [47]. In the present study, we followed the previous studies. First we denote

$x_{t,i}$ as i-th behavioral feature including eigenworm weights at time step t, and $y_i = [x_{1,i}, x_{2,i}, \cdots, x_{T,i}]^T$ as the time series of i-th behavioral feature from time steps 1 to T. The full-length behavioral data $Y = [y_1, y_2, \cdots, y_d]$ is $T \times d$ dimensional matrix where $d = 5$-dimension eigenworm weights. Then we lift the matrix Y into $(T - L + 1) \times Ld$-dimensional space of L contiguous delays.

$$\overline{Y}_L = \Phi_L(Y) = [Y_{L:T} \quad Y_{\mathrm{L}-1:T-1} \quad \cdots \quad Y_{1:T-L+1}] \tag{1}$$

(2)

Where $Y_{t1:t2}$ denotes the behavioral data from time steps $t1$ to $t2$. We used continuous time embedding with L=10 steps (corresponding to 2 sec) as the embedding time. After embedding, data from all individual animals were concatenated and independent component analysis (ICA) was performed using the Fast ICA method [81]. and embedded into a m-dimensional subspace Z_m,

$$Z_m = \overline{Y}_L \Gamma_m \tag{3}$$

where Γ_m is the $Ld \times m$ matrix for ICA dimensional reduction. In this case, time series of postures were finally embedded to $m = 5$-dimensional space.

For the analysis of the virtual behavioral data generated by the MDN-RNN, the time-delayed embedding was compared with the real animal data. Similar to above, a continuous time embedding of 2 seconds was performed. For the dimensionality reduction, we used a mapping $(\Phi_L(Y), \Gamma_m)$ that was obtained with real animals.

By applying time-delay embedding, generally we aimed at finding out 2 dimensions (forward modes) which activates during *C. elegans* forward movement, and 2 dimensions (reverse modes) which activates during its reverse movement. We refer to the subspace parameterized by the 2 forward modes as forward subspace, and similarly we define the reverse subspace.

1.4.3 Implementation of neural network models

Implementation and training of the MDN-RNN and RNN model

The output of MDN-RNN consists of parameters of each Gaussian distribution, which are centers (μ) and diagonal covariance matrix (Σ) for each Gaussian component, as well as weight (π) for each Gaussian distribution. The mathematical representation of the outputs is shown in equation (4) – (5), where K is the number of Gaussian components, x_t is the behavioral state at time t, h_t is the hidden state of the RNN at time t, and s_t is the stimulus at time t (0 or 1). The RNN consisted of three layers and LSTM was used. Each RNN layer consists of 128 neurons.

$$p(x_{t+1}|x_t, h_t, s_t) = \sum_{k=1}^{K} \pi_k(x_t, h_t, s_t)\mathcal{N}(x_{t+1}|\,\mu(x_t, h_t, s_t)\,, \Sigma(x_t, h_t, s_t)) \tag{4}$$

$$\sum_{k=1}^{K} \boldsymbol{\pi_k(x_t, h_t, s_t) = 1} \tag{5}$$

The model was implemented based on the Python and PyTorch [82] framework. Given the past and future pair ($X_{predicted}$, X_{real}), training was carried out by minimizing the negative log-likelihood. Parameter optimization was carried out using RAdam [83], which was claimed to be robust to learning rate change and able to eliminate the necessity of learning rate warmup.

For the inference, we sampled the next behavioral state from the mixture of Gaussians. First, the component was categorically sampled according to weight π, followed by sampling from the selected Gaussian distribution.

MDN-RNN takes 10-dimensional inputs: a1, a2, a3, a4, a5, angular velocity, velocity, direction (cosine and sine) and light stimulus information. The output of the model is designed to be a Gaussian mixture model which consists of K components of 9-dimensional Gaussian distributions to determine the probability

of a1, a2, a3, a4, a5, angular velocity, speed, direction (cosine and sine). To simplify the model, the covariance matrices were restricted to diagonal matrices. Therefore, the output of the network was a vector of K+K×D+K×D dimensions where K is the number of Gaussian distributions and D is the dimension of the next behavioral state, which is 9. The first K output neurons define the weights of each Gaussian distribution. Next K×D values represent the mean of each distribution and the last K×D values define the variance in each dimension of the distribution.

To test the role of MDN, we trained an RNN-based model, which only differed from the MDN-RNN model by the size of the MDN layer so that the output of this RNN-based model can be directly interpreted as behavioral state values of the next time step. We optimized the RNN-based model with MSE error until the same early-stopping condition with the MDN-RNN model was met. We saved the first 2000 time steps of prediction data and estimated the distribution that the RNN-based model gives, using various initialization conditions.

Quantification of the similarity between real behavioral data and generated behavioral data with Kullback–Leibler Divergence

Kullback-Leibler (KL) divergence measures the difference between two probability distributions. By treating the dataset as an i.i.d sample from the true distribution (of *C. elegans* behavioral states) and drawing random samples from the model as an i.i.d sample from the learned distribution, we utilized KL divergence to evaluate the likelihood of the model.

Admittedly, KL divergence does not directly measure the difference between the real and learned dynamics which drives the *C. elegans* behavior states and generates the distribution, and it measures the difference of the resulted distributions instead. Despite this limitation, KL divergence should still be considered as a valid quantitative measurement of the model quality.

Analysis of disentangled representation in MDN-RNN

To analyze the internal representation of MDN-RNN, we collected generated virtual behavioral data. We simulated 10 episodes and stored the behavioral states, stimulus, and parameters for mixture Gaussian distribution. Each episode consists of 10000 steps. The generated data were used to analyze the histogram for each feature of each component, to analyze the behavioral patterns by embedding them in the behavior map, and to analyze the dynamics in the behavioral state space created by the time-delay embedding.

The embedding of the behavior map was made by using a function of the openTSNE package. OpenTSNE was also used for adding simulation results to the behavior map created with the actual *C. elegans* behavior.

The conversion to behavioral state space was done by performing time-delay embedding under the same conditions as used in the actual *C. elegans* behavioral analysis and ICA.

1.5 Result

1.5.1 Acquisition and basic quantification of *C. elegans* behavior

In this study, *C. elegans* was selected as a target to apply the new model I proposed. First, to build a virtual model of *C. elegans*, we collected behavioral data of freely moving *C. elegans* using a high throughput behavioral assay developed in a previous study [78]. We placed worms on an agar surface and video-recorded their behavior at 5 fps with and without random optogenetic activation via channelrhodopsin (a light-activated ion channel) of sensory ASH neurons. ASH neurons are major nociceptive sensory neurons that mediate reversal responses (backward movement) to noxious stimuli including nose touch, heavy metals, and alkaloids [84]. We optogenetically activated these neurons via illumination to obtain information of sensorimotor responses.

Next, the behavioral sequence of *C. elegans* was quantified at each time point across 9 variables: 5 variables (a1 ~ a5) representing posture, and 4 variables representing velocity, angular velocity, and sine and cosine of head direction (Fig. 1-7 and Table 1-1). Eigenworm coefficients a1 through a5 were used for quantifying the body shape of *C. elegans* by conventional methods[73], while the centroid of the body was used for quantification of the velocity and angular velocity of each worm (see Methods and table).

To investigate the behavioral components and sequence structure, we first aimed to categorize behavioral patterns. We embedded behavioral data into 2 dimensions in an unsupervised manner by using PCA and t-SNE (Fig. 1-5). Embedded behavioral states formed distinct groups which were found to correspond to different behavioral patterns; forward, reverse and pause. Based on velocity and angular velocity, we were able to interpret these patterns and thus create a behavior map (Fig.1-8 A).

We also quantified and visualized the dynamics of *C. elegans* behavior by applying the time-delay embedding method which makes us able to extract dynamic properties of *C. elegans* behavior (Fig. 1-6; Ahamed et al. [47]). Visualization in two dimensions confirmed that different dynamics underlie the forward and backward behaviors (Fig. 1-8 B and C). We used these features to test how well the virtual models of *C. elegans* could replicate actual behavior as described in the next section.

1.5.2 A generative neural network model can successfully reproduce animal behavior

Training the MDN-RNN model of behavior

To simultaneously model the sequential and stochastic characteristics of behavior, which is the first goal of this research, I utilized the MDN-RNN model. This model is a RNN model combined with an MDN as an output layer (Fig. 1-3A). The RNN stores the information of previous inputs (behavioral dynamics), and this feature makes it capable of representing and predicting sequential data. The MDN transforms the values given by the RNN into a mixture of Gaussians (i.e. mixing the weights of each component of the mixture with the mean and covariance vectors). This enables the model to handle the stochastic features as probability distributions. MDN is an excellent way to model the data, especially for multistate and/or stochastic phenomena, including animal behavior (Fig. 1-3B). Since behavior is sequential and stochastic, MDN-RNNs is a powerful tool for modeling this type of biological phenomena.

In practice, I designed the MDN-RNN model to be composed of a 3-layer RNN followed by a 1-layer MDN (Fig. 1-3C). In this model, the MDN-RNN takes the behavioral state and stimulus information of the sensory ASH neuron for the past 20 s as input, and predicts the probability distribution of the behavioral state of the next step, which is a mixture of Gaussian distributions in nine dimensions. In the MDN-RNN simulator, the behavioral state of the next step is sampled from the Gaussian mixture distribution. First, an index of a Gaussian distribution is selected, according to the weight of each distribution (categorical sampling). Next, the value of the behavioral state vector is selected from the Gaussian distribution for the selected index, defined by the mean and variance of the distribution, which provides an output of the network. Hyperparameter search for the number of Gaussian components K was performed by minimizing negative log-likelihood (Fig. 1-9). I selected K=10 because the model trained with this value effectively disentangled the dynamics underlying the behavior (shown in the following section).

Qualitative validation of behavioral model

I trained the proposed model using the behavioral data of *C. elegans*. The behavioral patterns generated by the model were very similar to those of actual *C. elegans* (Fig. 1-10 A and B). The trained model was able to accurately represent the behavioral patterns during forward and backward movement. These results

suggest that MDN-RNN can represent multiple behavioral states in a single model, i.e., it has multiple internal models. To study this, I statistically analyzed the behavior generated from MDN-RNN. The histogram of velocity shows that the MDN-RNN generates both forward and backward behavior in a single model (Fig. 1-10 C). These results give support for the hypothesis that a mixed Gaussian distribution, the output of the MDN layer, enables the representation of multiple behavioral patterns in a single model.

Furthermore, I analyzed the dynamics behind the behavior using the time-delay embedding method. I found that the represented dynamics were very similar between actual *C. elegans* behavior and behavior generated by MDN-RNN (Fig 1-10 D to G), suggesting that the MDN-RNN model successfully learned the dynamics of the behavior.

Quantitative evaluation of behavioral model

For quantitative evaluation of the performance of the MDN-RNN and RNN model, I calculated Kullback-Leibler (KL) divergence between random samples of real *C. elegans* behavioral data, and between real data and generated data (Table 1-2). As a preparation, I split real *C. elegans* behavioral data into 10 parts (10 folds) and obtained 10 pairs by leaving one-fold out recursively. I also generated two behavioral data sets, each with the same size as one-fold defined above, using the MDN-RNN and RNN model respectively, and obtained 10 pairs for each model by replacing the 1-fold real data part with the generated data. After that, I estimated the sampling error with KL divergence of the real data pairs. I then estimated the model error, with KL divergence between the generated data and the real data. Note that for all KL divergence calculations, I treated the 9-fold part of each pair as true distribution, and the 1-fold part as sample distribution. Also note that sine direction and cosine direction are considered less important than other features, since the direction of worm movement tends to distribute uniformly in the long term. Thus, I excluded sine direction and cosine direction from the data set, in all KL divergence calculations. Apart from calculations with raw features, I also applied similar calculations to data embedded with t-SNE, so that quantification and visualization of the model error can be achieved simultaneously (Table 1-2 and Fig. 1-11). From these results, it was confirmed that the distribution of behavioral features generated from MDN-RNN were closer to the actual behavior of *C. elegans* than those

from RNN. However, it is not the same level as that of actual behavior of *C. elegans*, and there is still room for improvement.

As a final test for the accuracy of the MDN-RNN behavioral simulator, I validated the capability of the simulator to reproduce the behavioral response to sensory stimulation. As described above, the MDN-RNN model also takes stimulus information as an input. I quantified the frequency of reversal behaviors by delivering inputs that hypothetically stimulated ASH neurons. In real *C. elegans*, stimulation of ASH neurons has been shown to increase the frequency of reversal behavior. In this model, also, the frequency of reversal behavior increased in response to stimulation of ASH neurons (Fig. 1-12), indicating that the model learned the behavioral response of *C. elegans* to ASH neuron stimulation.

1.5.3 MDN-RNN disentangles behavioral patterns

Next, in this section, the process of the MDN-RNN model recapitulates behavioral sequences of *C. elegans* was further examined. Representation learning is a powerful feature of deep generative models. Deep generative models learn a meaningful representation in the model so that the data of interest can be generated with accuracy. Here, I will show that the MDN-RNN performs this disentangled representation learning and decomposes a behavior into different behavioral patterns. For the sake of convenience, I will hereafter use the term component to refer to each Gaussian distribution. Namely, a Gaussian mixture distribution as the MDN-RNN output consists of K (=10) components. As described above, this MDN-RNN simulator selects one component at each time point, and behavioral variables are randomly sampled according to the Gaussian distribution of the component.

First, I statistically analyzed the distribution of the behavioral variables in each component using a generated 10000-step prediction. I successfully found apparent differences in the distribution of velocity and angular velocity between the 10 components: reversal behavior was represented by components 0 and 1, pause behavior was represented by component 2, transition from forward to reverse was represented by components 3 and 4, and forward behavior was represented by the group of components 5 to 9 (Fig. 1-13 A and B). Among

the components representing forward behavior, component 9 represented mainly fast forward behavior. These results suggest a disentangled representation of behavioral patterns per component.

This result is consistent with the claim of Ellefsen et al. [77] that different components can be studied to model different possible future sequences. To clarify the differences between each component better, I represented the generated behavior in the behavior map by adding data created from actual *C. elegans* behavior. The results confirmed that the behavioral states sampled from the same component are close to each other in the 2-dimension latent space and hence occupy continuous areas, whereas those sampled from different components are relatively separated (Fig. 1-13C). This result is consistent with the previous claim that different components represent different behavioral patterns.

Furthermore, I mapped the dynamics of each component to the dynamics of actual behavior in the time-delayed embedded space, considering that each component may represent different dynamics (Fig 1-13, D and E). As a result, I found that the dynamics of each component corresponded nicely to a subset of dynamics in the time-delay embedded space.

Finally, to determine what types of behavioral features were represented by each component, I observed the behavior when each component was forcibly selected in a continuous manner. The MDN-RNN simulator performs random sampling of components according to the probability, and then extracts a sample from the distribution given by the component. However, in this test I tried to constantly sample from a specific component of the Gaussian mixture distribution. Surprisingly, I found that each component itself can generate a fluent series of movements and that the patterns of motion differed between components.

These results show that by modeling probabilistic time series data with an MDN-RNN, the dynamics of the system can be disentangled by the internal representation of the MDN-RNN. This means that the probabilistic generative model learns and internally represents the structure of the system during the training process of the time series prediction task. In this study, I treated the behavior of *C. elegans* as an example of probabilistic evolving time series of biological phenomena. This approach could be used for not only behavior, but also other types of data sets including neural activity dynamics.

1.6 Discussion

1.6.1 Disentanglement of behavioral dynamics by representation learning

An essential advantage of this approach is the automatic classification of animal behavior through representation learning of deep generative models. I applied a powerful feature of deep generative models, representation learning, to analyze animal behavior. A deep generative model is a machine learning method that learns the underlying structure of the data being generated, and the model learns a mechanism by which it can generate realistic data [85]. In the process, the model casts the underlying structure of the data and organizes the representation into features with different properties. It is known that MDN-RNNs models learn different dynamics in a disentangled representation as a time-evolving Gaussian mixture [77]. More precisely, it is known that different Gaussian components in the MDN layer have two complementary roles: to separately model the different events governed by different dynamics, and to separately model the different stochastic events [77]. In this research, I successfully modeled multiple behaviors governed by stochastic elements and different dynamics of animal behavior using these features. I showed that the complex dynamics are unraveled and self-organized for each of the different components in the learned model.

1.6.2 Applicability of sequential mixture density neural network

The modeling methods developed in this dissertation are not limited to *C. elegans* behavior. In fact, the model can be applied to behavior and neural activity of other animals. I showed that MDN-RNNs can be used to simulate *C. elegans* behavior with high accuracy. Technologies for behavioral quantification from time series images are rapidly developing [4]. Accordingly, behavioral states of various animals have been characterized, and human motion has been captured and analyzed quantitatively. Therefore, using variables from continuous animal behavior or human motion as inputs, I could potentially simulate the behaviors of many animals, including humans, whose behavior is more complex than the worm's behavior analyzed in this study. One of the important properties of any method for analyzing behavior is that it can be easily scaled to the analysis of other animal species. The method I adopted can handle time series data in any format. Various

characteristics of animal behavior have been quantified so far, and recent developments in technology have made it possible to easily obtain skeletal data of human and animal body postures. The methods presented in this study are compatible with all of these methods and could be used as the first of the post-quantification steps of behavioral analyses.

Furthermore, the proposed method is not limited to behavioral data. In principle, this method can be applied to any stochastic sequential data in biology and can learn meaningful representation to generate those biological phenomena. Neural activity is one of the first targets for extension of this work as probabilistic time series data modeling. In our group, we have started modeling neural activity data using MDN-RNN. The probabilistic deep generative model is effective for both direct modeling of the phenomenon itself and parameter estimation of mechanistic models, and further applications are expected in the future.

1.6.3 Building the first complete computational model of *C. elegans*

In this study, I have succeeded in creating a "virtual nematode" that mimics the behavior and specific sensory responses of the nematode. By extending the results of this study, I expect to be able to create a virtual nematode whose neural network more closely resembles the actual neural connections of a nematode, and whose internal representation is more similar to that of an animal. By closing the gap between the data-driven generative model created in this research and the mechanistic model created based on the neural circuit structure and skeletal structure, it will be possible to create a generative model that is closer to the control mechanism of actual animal behavior. *C. elegans* is a particularly good model animal for this purpose. In this paper, I discuss the significance of creating a "virtual nematode" and ideas on how to create it.

We still do not understand the mechanisms by which the complex and diverse animal behaviors found in nature are generated. The main challenge for neuroscience is to understand how each neural activity is related to behavior and how it is involved in the generation mechanism. By creating virtual animals and reverse engineering animal brains, ethology can benefit in many ways. Describing the process of generating behavior

is the goal. Moreover, if we can predict which part of the generation process is abnormal and leads to psychiatric disorders, we can select effective targets for treatment. Furthermore, the mechanism of embodied control will be highly valuable in AI research, especially in robotics.

C. elegans is an excellent model organism for linking neural activity to behavior. The nervous system of *C. elegans* consists of 302 neurons, and all of their neural connections have been identified. With the technical development of high-speed microscopy and tracking systems, it is now possible to record neural activity during free-ranging behavior. Therefore, it is now possible to obtain both behavioral data and data on almost all neural activity in the head at the same time. With this background, I believe that it will be the first animal to be able to reproduce neural activity and behavior in silico, based on actual neural activity data and structural connections obtained.

At present, RNNs are reproducing the behavior of animals and processing information in the actual nervous system. I believe that it is possible to construct a neural network that is more biologically similar to an animal. Our group has previously developed a mechanistic model of *C. elegans* behavior, and we have already begun to integrate this mechanistic model with my data-drive model to create virtual nematodes that correspond to virtual neural activity and locomotion. The results of this study, which proposed a data-driven generative model, are very important as a foundation for research on creating such virtual animals, and I look forward to further development.

1.6.4 Real world application of behavior prediction

Prediction of animal behavior, including human behavior, has a huge impact on social application. For example, in the field of robotics, human behavior prediction models enable robots to anticipate how humans may react to the robots' actions. The same is true in the realm of automated driving. Estimating the probability distribution of how people behave and where they will be in a few seconds afterwards is an essential part of the technology to make automated driving work safely. Hence, probabilistic prediction of human behavior has a huge impact on the social application of robots and other applications.

1.7 Figures and Tables

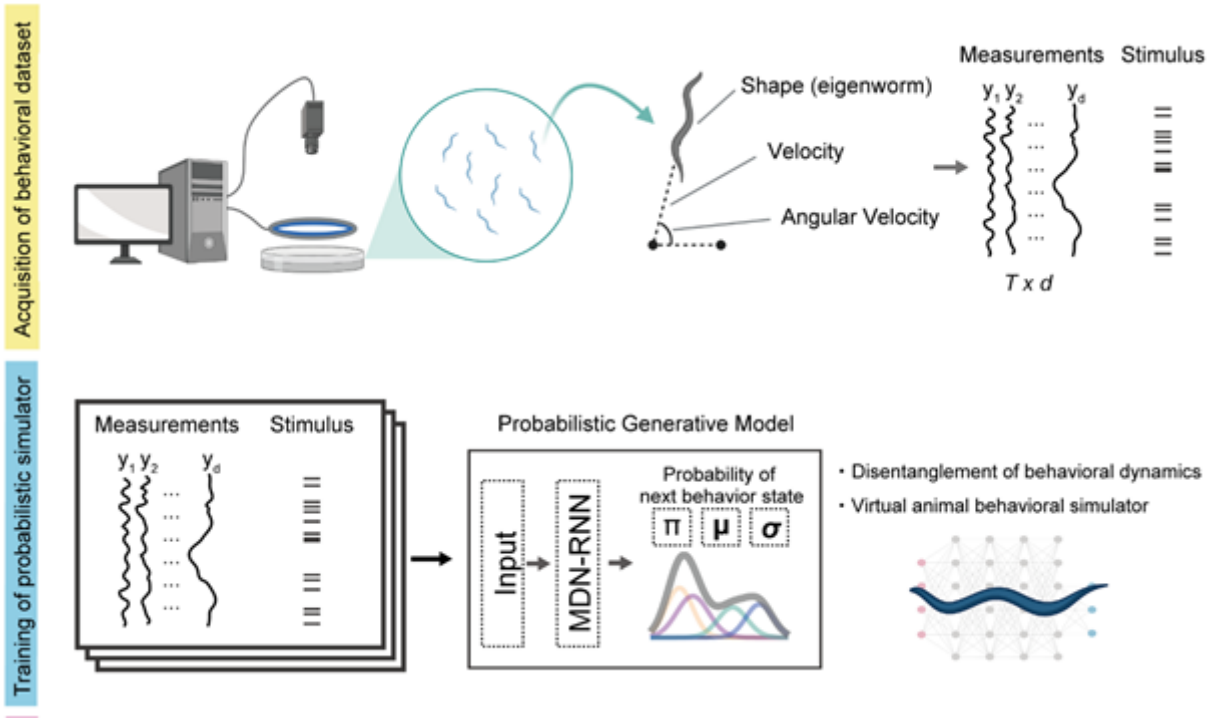


Fig. 1-1 Research paradigm.

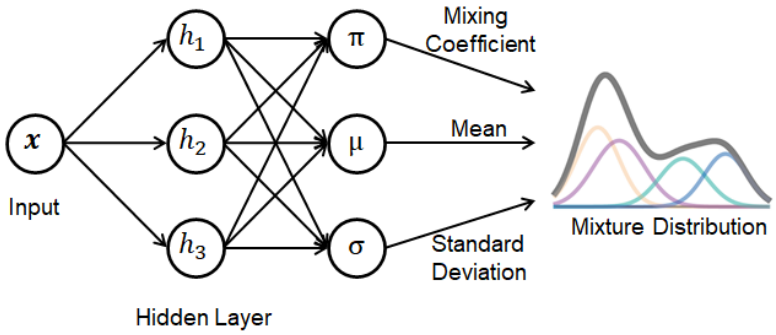


Fig. 1-2 Schematic illustration of mixture density network.

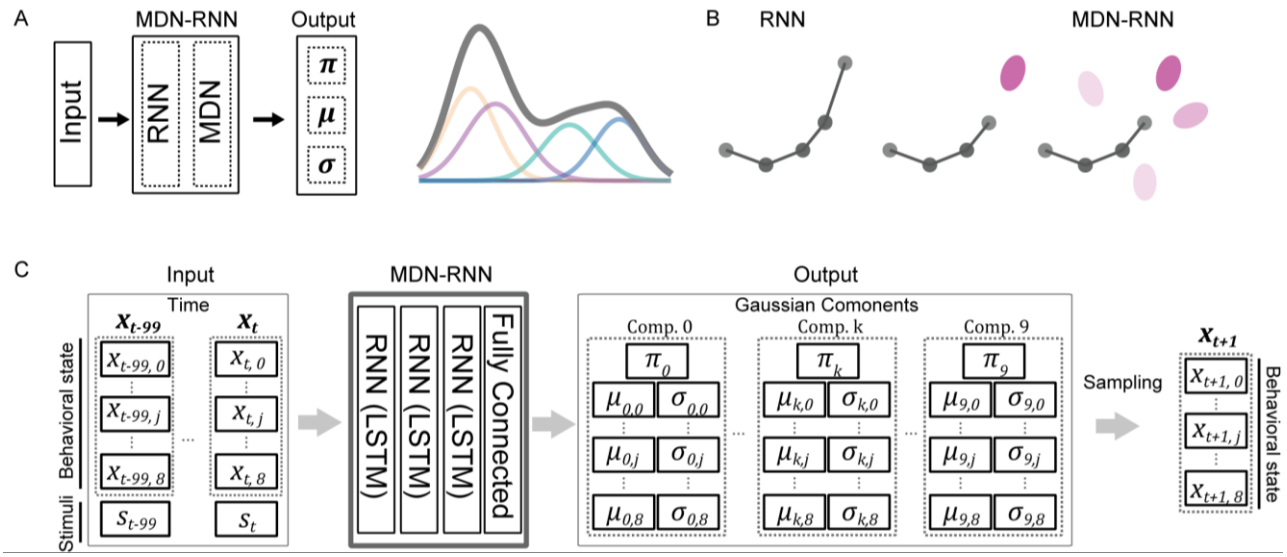


Fig. 1-3 Schematic illustration of mixture density recurrent neural networks (MDN-RNN).

(A) Schematic illustration of the MDN-RNN. The input is first processed by the RNN and then transformed into each parameter of the Gaussian mixture distribution by the MDN layer. (B) A conceptual diagram of the prediction method for time series data. The predicted values in the near future is deterministically output from RNN based on data of the past time points, which are shown as dots. On the other hand, MDN-RNN outputs the probability distribution of the next predicted values from the data of the past time steps, which is showed as colored ovals. (C) A detailed description of the MDN-RNN used in this study. The behavioral states of the past 100 steps are received as input, processed by the three-layer RNN, and then transformed by the MDN layer into each parameter of a Gaussian mixture distribution consisting of K Gaussian distributions, where x_t is the behavioral state at the time t, s_t is the stimuli at time t, and π_k, μ_k and σ_k is the parameters of the Gaussian mixture distribution.

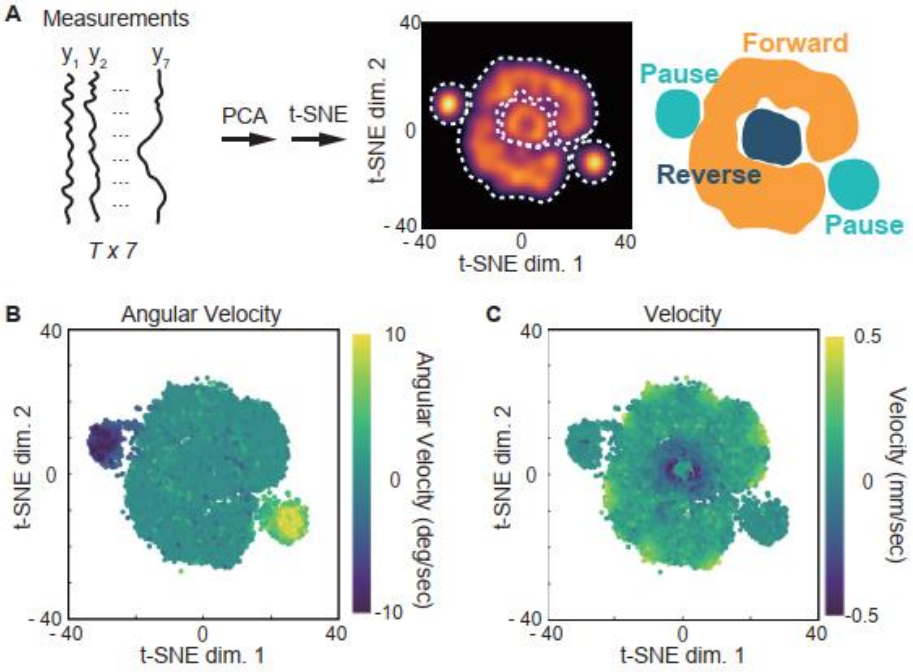


Fig. 1-5 (A) Schematic illustration of 2D behavior embedding. (B and C) Relationship between position on the behavior map and angular velocity and speed.

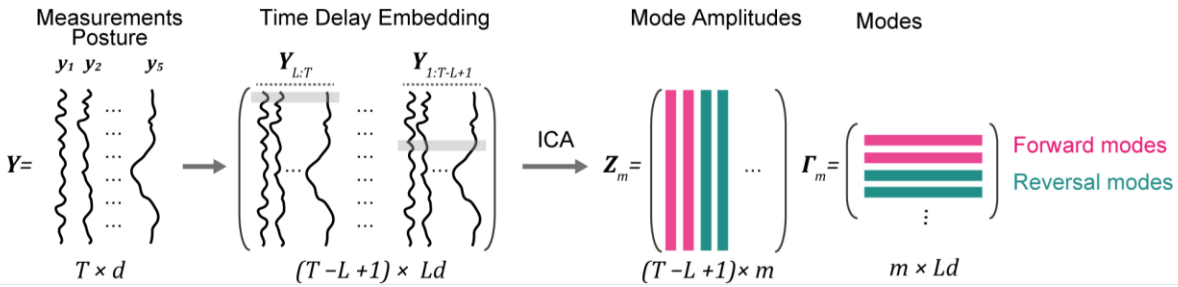


Fig. 1-6 Schematic illustration of behavioral dynamics analysis by time-delay embedding and independent component analysis (ICA). Time delay embedding was followed by ICA to extract the behavioral modes, each of which is dominated by different dynamics. These analyses were performed using only the information of the posture of *C. elegans*.

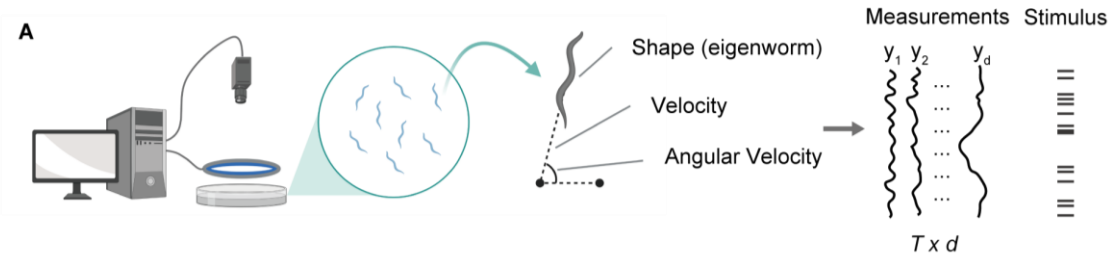


Fig. 1-7 Acquisition of the behavioral dataset. Schematic illustration of data acquisition and quantification process. Freely moving animals were video-recorded by a high-throughput assay system. Behavioral states are expressed in nine dimensions. The behavioral states were used to create a behavioral map and to analyze the dynamics using time-delay embedding.

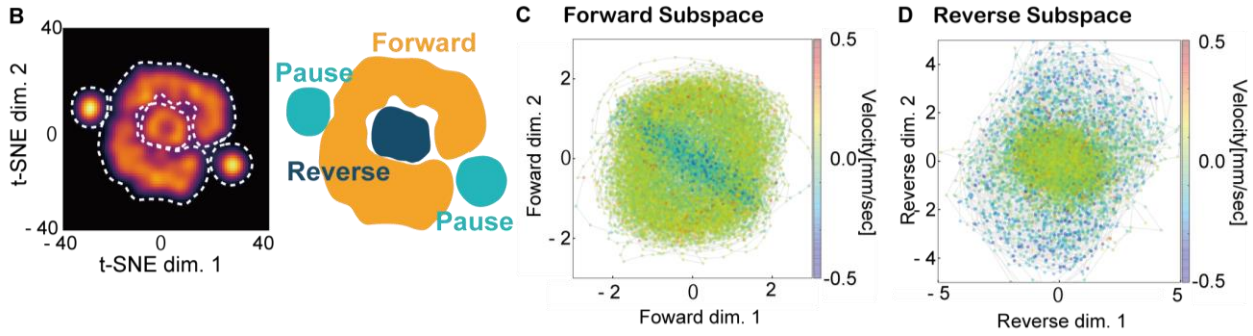


Figure 1-8. Quantification of the behavioral dataset. (B) Behavior map of freely moving C. elegans. Behavior is embedded in two dimensions and appears to be mainly affected by the value of the velocity and the value of the angular velocity. The value of the velocity corresponds to the forward and backward movement of the nematode, and when absolute value of the angular velocity is large, the animal remains in one place . (C and D) Visualization of the dynamics in a two-dimensional space. The dynamics changes significantly during forward motion (C) and reverse motion (D). Each time point is colored according to the velocity.

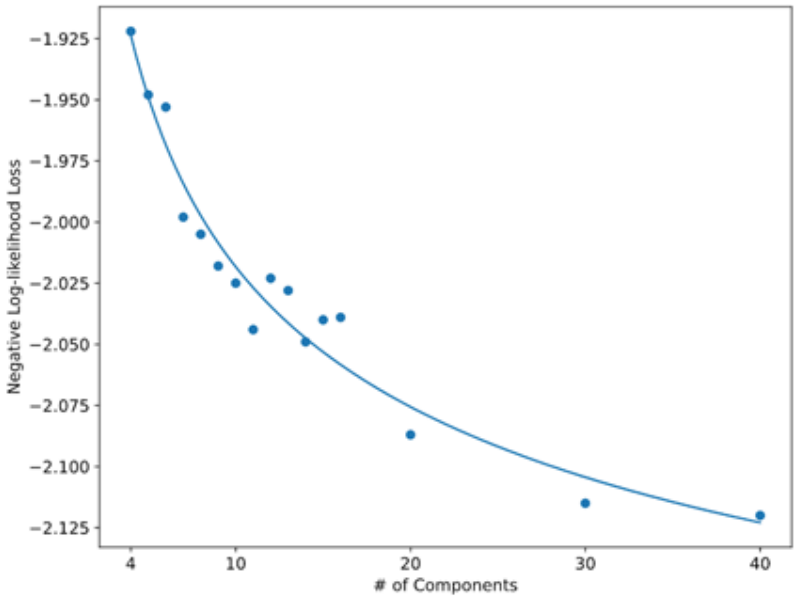


Fig. 1-9. Relationship between number of components and mixture density recurrent neural networks loss.

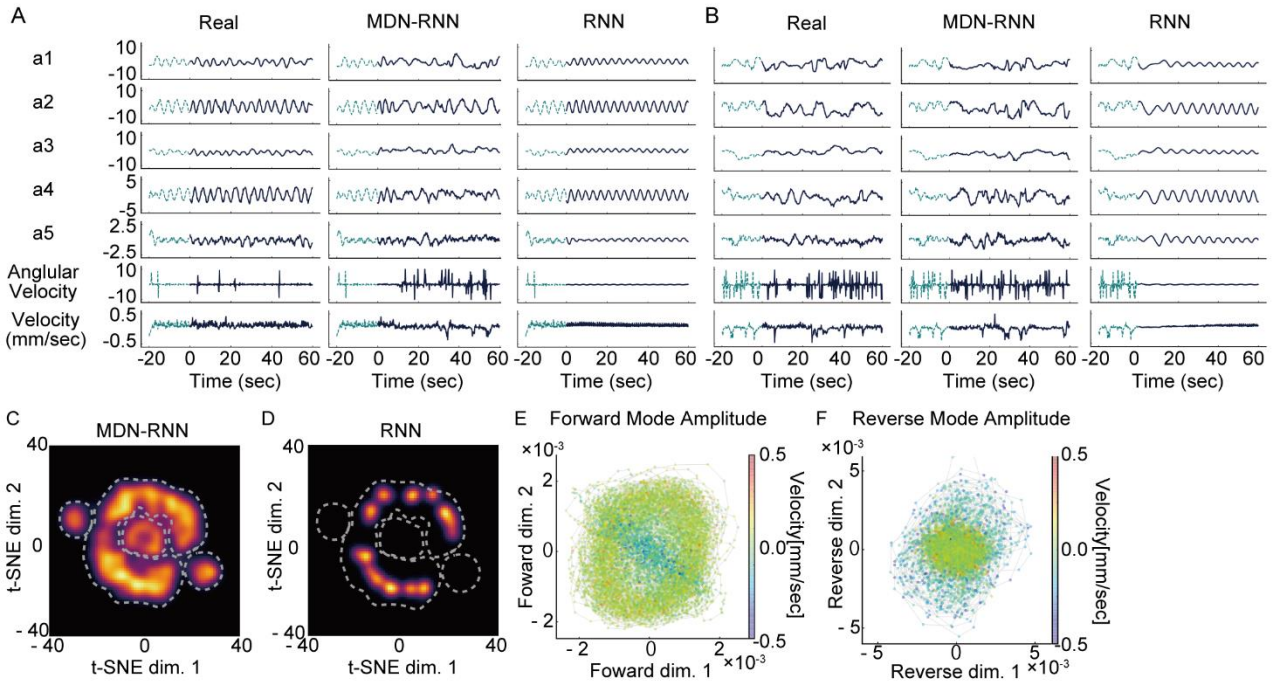


Fig. 1-10. A generative neural model can successfully reproduce the behavior of animals. (A and B) Comparison of the behaviors generated by the probabilistic generative models and the actual nematode behavior. The light blue area is the actual nematode behavior data used for initialization; the Real column shows the subsequent behavior of the real nematode; the MDN-RNN and RNN columns show the behavior of the virtual nematode generated by the trained simulator. A and B illustrate the results of initialization using different behaviors of real animals. (C and D) Comparison of MDN-RNN and RNN distributions on the behavior map. (E and F) Trajectory in a behavioral state space (E: forward and F: reversal).

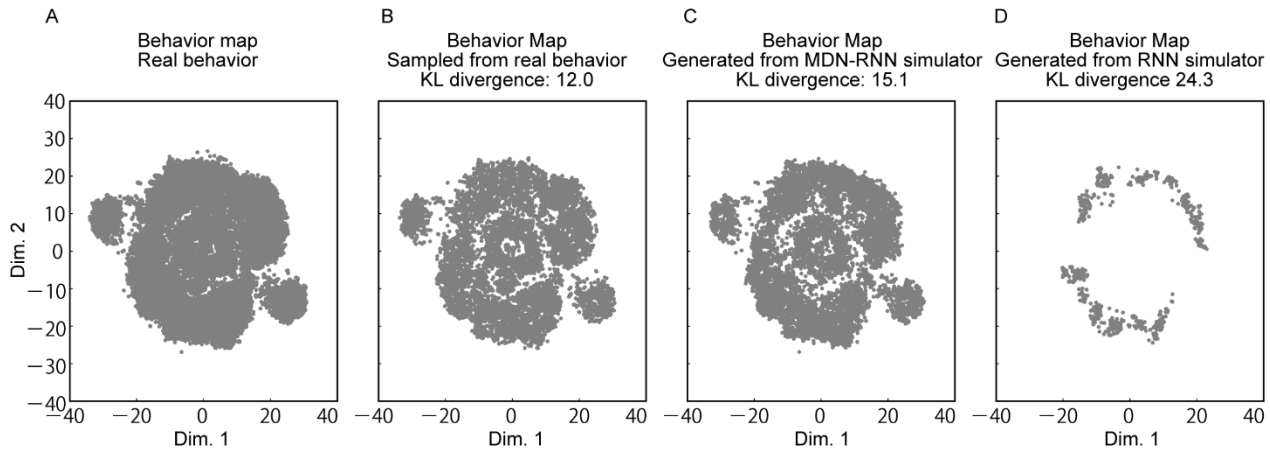


Fig 1-11. Qualitative evaluation of the accuracy of simulated behavior

(A) The behavior map embedding of the real *C. elegans* behavior. (B) - (D) Results of KL divergence comparison with actual *C. elegans* behavior shown in (A). (B) Comparison results of KL divergence with sampled actual *C. elegans* behavior. (C) Comparison results of KL divergence with the behavior generated from MDN-RNN. (D) Comparison results of KL divergence with the behavior generated from RNN.

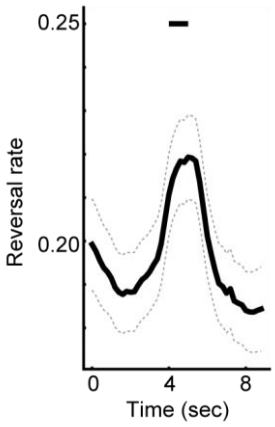


Fig 1-12. Response of the model to stimulus input. After 1 s of stimulation (indicated by the bar) the reversal rate increased. This indicates that the model animal responded to the stimulus input. The dashed lines represent the standard deviation.

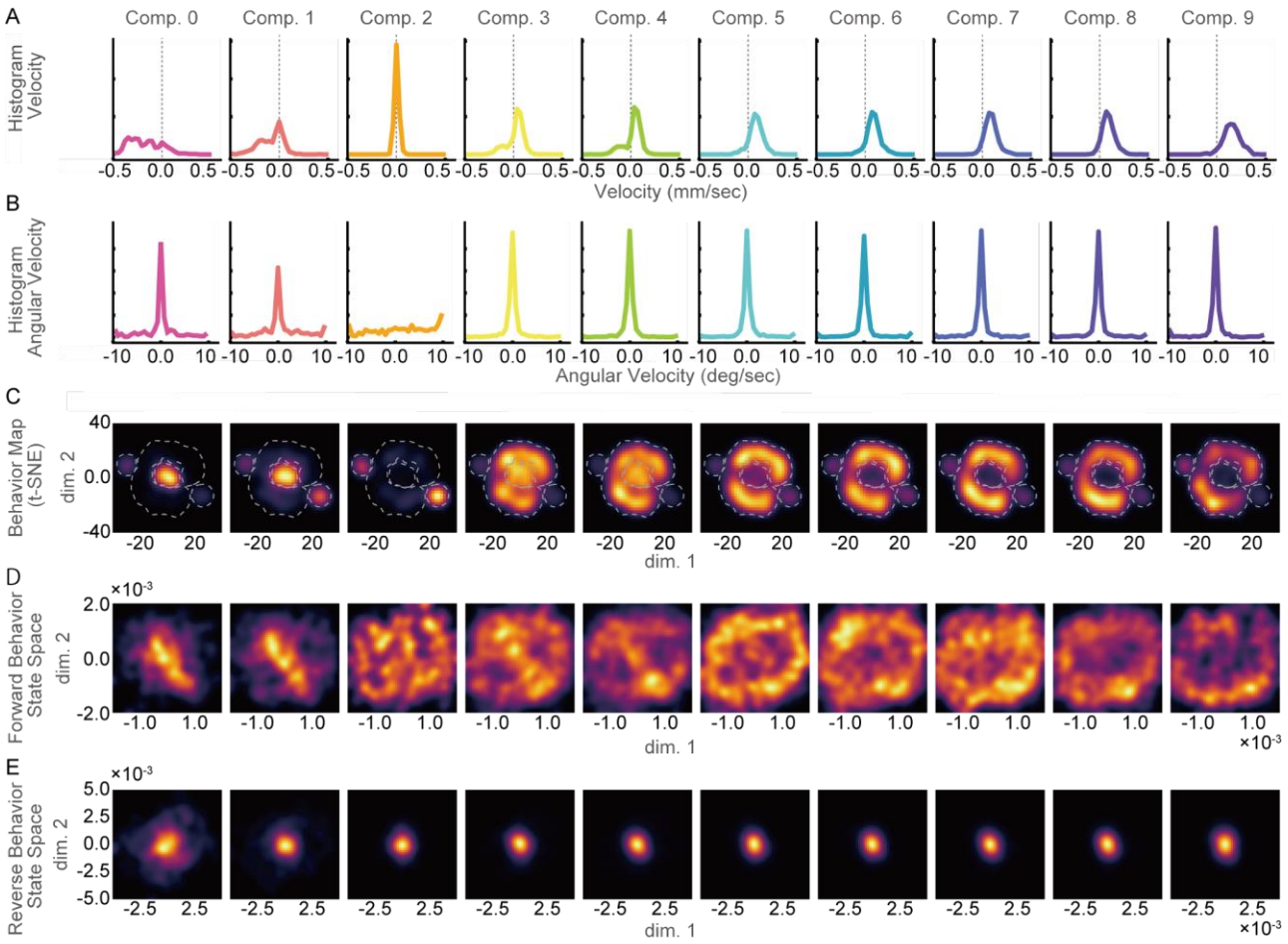


Fig. 1-13. Behavior dynamics is disentangled by the deep generative models. (A and B) Histogram of velocity (A) and angular velocity (B) for each behavioral state generated from each component. Reversal behaviors are mainly represented by components 1 and 2, while turn behaviors are represented by component 3. (C) Probability density on the behavior map for each component. (D and E) Trajectory in behavioral state space (D: forward, E: reversal) when each component is selected.

Table 1-1. Variables for the quantification of the behavioral state

	Description	Variable name in the code
Shape	Eigenworm component 1 weight	a1
	Eigenworm component 2 weight	a2
	Eigenworm component 3 weight	a3
	Eigenworm component 4 weight	a4
	Eigenworm component 5 weight	a5
Velocity	Angular velocity	AngularVeolcity
	Velocity	VelocityTailToHead
Direction	Sine of advance/retraction angle	Direction sine
	Cosine of advance/retraction angle	Direction cosine
Stimulus	Led on (1) of off (0)	led

Table 1-2. Quantification of sampling error and model error (average ± sem)

	KL divergence *with raw features*	KL divergence *with T-SNE 2d-embedded data*
Sampled real data	0.117±0.002	11.928±0.132
RNN	8.229±0.001	24.440±0.043
MDN-RNN	0.432±0.001	15.133±0.016

Chapter 2

Behavior control via reinforcement learning

2.1 Abbreviation

C. elegans	*Caenorharbitis elegans*
DQN	Deep Q Network
MDN-RNN	Mixture density network - recurrent neural network
NN:	Neural network
RL	Reinforcement learning

2.2 Introduction

Understanding animals' behavior is of great importance in various fields including but not limited to ethology, neuroscience, and robotics [58–60]. From a biological standpoint, replicating animal behavior is important for understanding how the nervous system controls an animal's body and how it processes sensory information and makes decisions to generate corresponding motion. From an engineering standpoint, animal-inspired robots are in heavy demand for various applications [60–62]. Although previous studies have been focused on extraction of behavior strategy in hypothesis-driven manner, recent advancement of behavior measuring and machine learning will make it possible to automatically extract the behavioral strategies. Therefore, the research aimed to replace the animal's behavioral strategy in a specific behavioral task with a machine.

In this research, we examined the effectiveness of using reinforcement learning (RL) for automation of controlling animal behaviors in the simulation environment developed in *Chapter 1* (Fig. 2-1). RL is a type of machine learning that is specialized for learning the policies for specific tasks [86]. Previous studies have suggested that mammals adopt an RL framework to decide their behavioral strategy [86–88]. By replacing animals' learning systems with machine learning systems, I aimed to control virtual animal behaviors *ad arbitrium*. Because RL generates policies which achieve specific tasks, control of animals is accomplished automatically and systematically; without the need to specify how to manipulate animals.

2.3 Background and related research

2.3.1 Reinforcement learning

In this section, I will briefly describe the concept of reinforcement learning. Reinforcement learning is a type of machine learning in which machines try to learn the best policy to solve a target task automatically during the process of trial and error. In the reinforcement learning setting, the agent (machine) and the surrounding environment interact with each other (Fig. 2-2). The following three steps constitute a reinforcement learning trial; the environment presents the *state* to the agent, the agent decides the *action*, and the environment updates the state and also passes the *reward* to the agent depending on the action. The goal of the training is that agents learn the policy which can obtain rewards as much as possible from the environment. There are several algorithms for the update (learning) of the policy. The algorithm used in this study is detailed in the methods section.

2.3.2 Replicating animal behavior via robots

Replicating animal behavior and body control systems have great benefits on robotics. Several works have tried to replicate the animal body control system by building mechanist models[61] and/or using machine learning methods such as reinforcement learning models [60].

However, few works have been done on autonomous extraction of behavioral strategy and its application to robots. Since many animal-inspired robots have been introduced into practical use, it will become increasingly important to imitate and deploy behavioral control strategies which adapt to the surrounding environment from animals themselves. In this manner, methods for extracting autonomous behavioral strategies are needed.

2.4 Method

2.4.1 Problem setting

Hypothetical navigation task was designed to test the usefulness of reinforcement learning in controlling animal behavior and extracting behavioral strategies (Fig. 2-3, 2-4). In reinforcement learning, the problem setting can be divided into environment and agent. The interaction between environment and agent is shown below. The environment presents the current state, and the agent decides the action based on the current state and its own policy. The environment then receives the action from the agent and returns the next state and reward to the agent. The details of environment and agent are described in the next section.

Environment (Navigation task)

The environment in this task consists of the goal position and behavior state of the animal (*C. elegans*) in simulation space, and returns the rewards based on the agent's position. State is composed of the direction and speed of the worm and its posture as values of eigenworm (a1 and a2). Direction of the worm was calculated as a difference of angles between the direction of the goal point and moving direction of the worm in the following equation,

$$direction = dirction_{goal} - direction_{worm}$$

where as, $direction_{goal}$ is the angle toward the goal point from the position of *C. elegans* and $direction_{worm}$ is the heading angle of *C. elegans* in the simulation environment. Speed is represented as velocity, and posture has values of eigenworm a1 and a2 (refer to the behavioral quantification section in the *Chapter 1* and Table 2-1 for explanation of these variables).

Table 2-1 Variables for the reinforcement learning environment state

State Name	Description	Variable name in the code
Eigenworm 1	Posture variable 1	a1
Eigenworm 2	Posture variable 2	a2
Direction	Difference between goal direction and worm direction	angle
Speed	Velocity of the worm	speed

The simulator of the environment receives the previous state and action and returns the reward and next state. The simulation of behavior conducted based on the MDN-RNN model, which was created in *Chapter 1*.

The task was designed to control the behavior of *C. elegans*: navigating toward the goal point (Fig. 2-4). Reward was defined as the decrease of distance (pixels) from the goal point (if *C. elegans* approached the goal, it obtained a positive reward) and an additional bonus reward of 1000 was obtained for reaching the goal. The reward was calculated by the following equation.

$$Reward = 1000 \; if \; reached \; threshold \; distance \; from \; goal; \; else \; (dist_{t-1} - dist_t)$$

$dist_t$ and $dist_{t-1}$ represent distance to the goal point from the position of the worm in time point t and t-1, respectively, and so indicate the change of distance toward the goal point.

Agents

In this task, the agent is the optogenetic controller, which is designed to obtain the greatest possible rewards through the RL scheme. The binary action of turning the lights on/off activates the ASH neurons optogenetically (Table 2-2). Agents decide the action based on the policy π. The objective of the training is to find the *policy* π to maximize the sum of future rewards.

Table 2-2 Variables for the reinforcement learning environment state

Action name	Description	Variable name in the code
Stimulus	Optogenetic stimulation of ASH neurons	led

2.4.2 Agent model

Q-learning

As mentioned in the previous section, the purpose of learning is to obtain the policy which makes the agent to obtain high reward from the environment. An effective way to learn such policy is to formulate it as the function of the discounted reward to each of the state-action pairs, which is called q-values,

$$Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\}$$

By setting $\pi = \pi^*$, I can obtain the optimal policy π^* by learning the q-values for every state-action pairs. Such a learning method is called Q-learning.

Once the q-values are learned, the optimal policy at a given state is to choose the action with highest value with probability $(1 - \varepsilon)$ where ε is the exploration factor. This method of action selection is called the greedy method.

Deep Q-learning (DQN)

In the case where the task has a large state and action space, learning q-values (or creating q-table) becomes intractable. DQN was introduced to address this problem by using a deep neural network to estimate the q-values from the state [89]. In DQN, policy is represented as a multi-layer neural network which takes $State_t$ as the input and outputs q-values for each possible action. I adopted Deep Q network as a model of reinforcement learning, in which the NN reduces temporal difference error through training with Q-learning to decide action in RL. NN suits for dealing with serial states because NN can represent complex nonlinear

functions. Since *C. elegans* behavioral state is in a continuous space, I considered that the NN is suitable for this RL framework.

2.4.3 Implementation

NN in the RL model has the input layer of 4 components, which corresponds to the number of input variables, 2 hidden layers, each of which have 10 components and an output layer of 2 components corresponding to the number of output variables. Activation functions of hidden layers were ReLU [90]. Adam (a stochastic gradient based optimization method)[91] was used as the solver for weight optimization, and behavior distribution was decided by ε-greedy policy. I implemented the RL model using the Python and machine learning library scikit-learn [92].

2.4.4 Training

To train an agent to learn a policy for achieving tasks, I set hyper-parameters. In my model, ε (probability of acting randomly) was 0.05 for ε-greedy policy, γ (discount factor) was 0.95, max epochs were 1000 and, in each epoch, max steps were 400. To stabilize the training results, I used Experience Replay, in which action history was stored temporarily and then sampled to be used as training data. Buffer size (number of stored data) was 1024 and batch size (number of data used per single training) was 32. Each training session was conducted five times, each with a different random seed.

Trained models were also tested to evaluate the performance of models or to reveal the policy learned by the trained models. When the models were tested, ε was 0.0001 and the parameters of the neural network were not updated. Models were tested in an environment where their start points were close (randomly chosen within 0.2 – 0.5 mm far) from the central point and they could end their epoch and received a bonus reward when they reached the goal area, and in an environment in which their start points were far (randomly chosen within 0.2 – 1.0 mm from the central point) and they could not end their epoch until 400 steps had passed,

although the rewards grew larger as they approached the center of the arena (where the goal was located). The former setting was used to evaluate the model ability to obtain rewards and navigate toward the goal point, and the latter setting was used to show how worms accumulated near the goal point when they were stimulated by the learned neural activation sequence. The other settings were the same as those in the training session.

2.4.5 Statistical analysis

Statistical analysis was performed in the Chapter 2 using the SciPy library in python. The Brunner-Munzel test, a nonparametric method, was used to test for differences in obtained rewards between the two groups, and the chi-square test, a nonparametric method, was used to test for differences in goal rates between the two groups. All *P* values are listed in the figure legends.

2.5 Result

In this chapter, I aimed to automatically extract the behavior strategy of *C. elegans* via machine learning. By automatically finding the policy to make *C. elegans* perform a specific behavior (e.g. make worm dance), I tried to extract the possible behavioral strategies. I adopted RL which is a (machine learning) framework suitable for automatically obtaining the best policy to manipulate animal behaviors to perform a predefined task automatically [86].

In an RL framework, an agent, environment, action, state, and reward need to be defined. In this research, the agent is the optogenetic controller, the environment is the position, posture and internal state of the virtual worm, the action is optogenetic activation of the ASH neuron, and the reward is pre-defined for each task (Fig. 2-3). I adopted a Deep Q Network (DQN) as a model of reinforcement learning because DQNs can learn the appropriate policy in a series of continuous states, which are the posture and behavior of worms in this study[93].

I designed a task whose objective is to navigate a worm toward a goal point. This task corresponds to the organism's taxis behavior. The goal point was the center point of the test arena and the start points of virtual worms were randomly selected to be within a certain distance of the goal point. To complete this objective, I designed the reward to be a decrease of distance (pixels) from the goal point (if the virtual worm approaches the goal, it gains a positive reward) and a bonus reward for reaching the goal (Fig. 2-4). By this configuration of rewards, the agent is expected to learn a stimulation sequence of ASH which shortens the distance of the animal to the goal point. The goal bonus reward is so much larger than the reward for change of distance that the total reward is mostly affected by the goal bonus reward.

2.5.1 Training result

As a result of training, I succeeded in automatically controlling the virtual worms to navigate toward goal points in a simulated environment. The rewards increased over the course of epochs when the simulation

sequence was trained in a reinforcement learning framework (Fig. 2-5A). To assess the performance of the trained model, I used the trained model to stimulate the nematode and compared its behavior to that caused by the random stimulation. Neural stimuli in a random sequence with a 50% irradiation rate were used as comparators because the initial condition for reinforcement learning is a 50% irradiation rate. Compared to the random case, the virtual worms in the learned group earned more rewards, clustered near the goal, and had a higher goal rate (Fig. 2-5 B and C; Fig. 2-6).

2.5.2 Computational mechanism

Furthermore, I investigated the computational mechanisms that the automatically learned policy developed. To do so, I studied the relationship between the angle of the *C. elegans* to the goal and the stimulation rate. As a result, I found that an agent learned the policy to stimulate the neuron when virtual worms were not facing the goal point (Fig. 2-7). I concluded that through this policy, virtual worms are made to move ahead when they are facing the goal point and to change their movement direction via a reversal and turn when they are not facing the goal point. To verify this assumption, I made a toy model which stimulates the neuron when virtual worms do not face the goal, and which does not stimulate the neuron when facing the goal. This toy model performed better than random activation sequences and were similar to the trained sequences, indicating the effectiveness of the obtained policy (Fig. 2-5A). Interestingly, actual *C. elegans* uses a similar strategy called the pirouette mechanism for chemotaxis [94]. According to this mechanism, real worms turn more frequently when they are facing away from a chemoattractant. Other organisms and cells such as *E. coli* and sperm also use this kind of navigation strategy [95]. Thus, RL gives us universal and effective navigation policy of organisms as well as meaningful insights for understanding animal behavior by elucidating the behavioral strategy of existing animals.

2.6 Discussion

In this chapter the computational methods for extracting behavioral strategy from animal behavioral data was developed. In this task, I have succeeded in automatically extracting the pirouette strategy used by *C. elegans* during navigation by reinforcement learning. The extraction of behavioral strategies is beneficial not only for ethology but also for many other fields. In the following chapters, the usefulness of the method in ethology and its usefulness in peripheral fields such as brain machine interface and robotics are discussed.

2.6.1 Importance and future direction in ethology

Understanding the behavioral strategies of animals is an important question in ethology. In other words, it is important in ethology to clarify which information in the environment is used, what kind of computation is performed, and how the final output, behavior, is achieved. Now that behavioral measurement has become easier and behavioral data in large scale and naturalistic environments are available, it will be possible to find behavioral strategies that have not been discovered in the past by having machines automatically extract behavioral strategies.

Although the setting in this study is that the Agent receives the state and outputs the action of giving an aversive stimulus to the animal (activating the ASH sensory neurons), it may be more appropriate to control the motor command neuron as the action. It is known that the stimulation of ASH sensory neurons indirectly induces a retreat behavior, while the stimulation of motor command neurons directly induces a specific behavioral component. In this study, for simplicity, only backward behavior was targeted for action, but it is possible to target multiple behavioral elements, and I hope to address this issue in the future.

In this study, behavioral strategies were extracted on the virtual animal simulator created in Chapter 1. In the future, however, it will be necessary to verify the extracted behavioral strategies using actual animals. However, in the future, it will be necessary to validate the extracted behavioral strategies using real animals. For this purpose, it is necessary to develop an experimental system in which recognition of current state and

stimulation of neurons are performed in a closed loop manner. Such a closed-loop experimental system has been developed mainly for model organisms, and in recent years, it has become possible to adapt the system to free roaming animals in three dimensions. Future directions may include the integration of these experimental systems and the reinforcement learning control system developed in this research.

Research on neurostimulation such as neuromodulation for medical applications has been rapidly advancing in recent years. RL is one option for a controller of brain activity. Several studies have been carried out on monitoring neural activities and in turn controlling them using computational models and RL [96,97]. In bidirectional brain machine interface, nervous systems of patients were stimulated to exert specific functions such as normal movement [98]. However, the number of studies that aimed at controlling brain activity or behavior of living animals through neural stimulation patterns obtained by RL is still limiting. Future studies should target restoring human brain functions in patients with nerve damage or behavior of simpler animals *in vivo* by constructing a closed loop RL model. Therefore, application of the outcome of this study to real animals will enable us to make the neuromodulation process more accurate and systematic using the policies of controlling systems obtained via RL and to support behavior of patients to restore motor functions through generating optimal brain stimulus sequences automatically.

2.6.3 Application to robotics

One of the practical applications of this research is robotics. In recent years, robots that mimic the body control system of animals have begun to be proposed and used in the real world. These animal-shaped robots have succeeded in imitating the posture control of animals, but the behavioral strategies are still designed by human hands. In the future, it is expected that intelligent robots that mimic animal behavior, even down to the behavioral strategies, will be needed. In this context, methods for extracting behavioral strategies for accomplishing specific tasks, as achieved in this study, are required. I believe that the proposed method can serve as a foundation in this context.

2.7 Figures

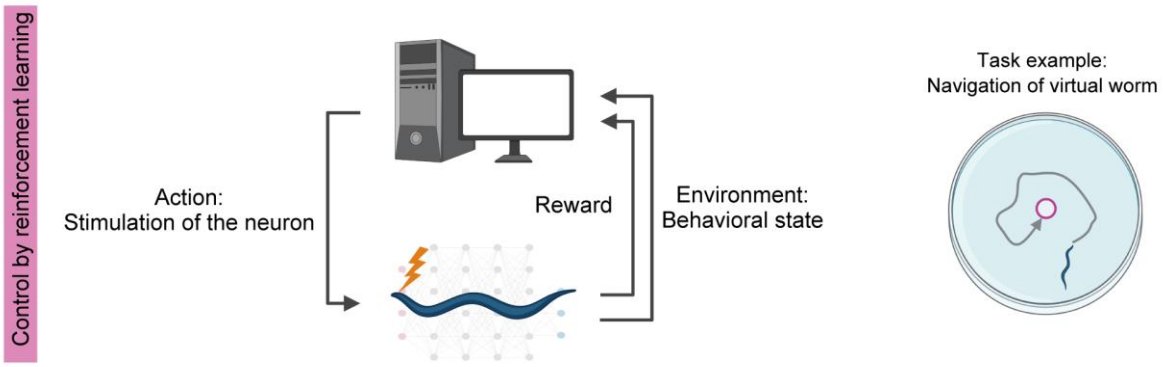


Fig. 2-1 Research paradigm.

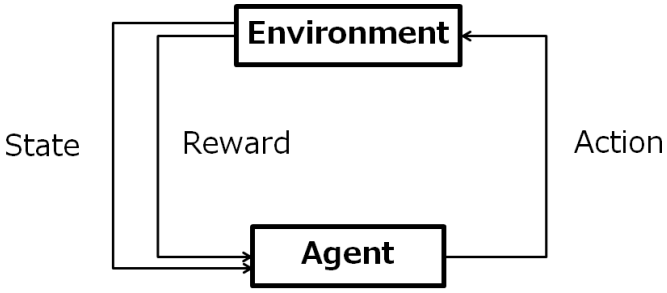


Fig. 2-2 Schematic illustration of reinforcement learning.

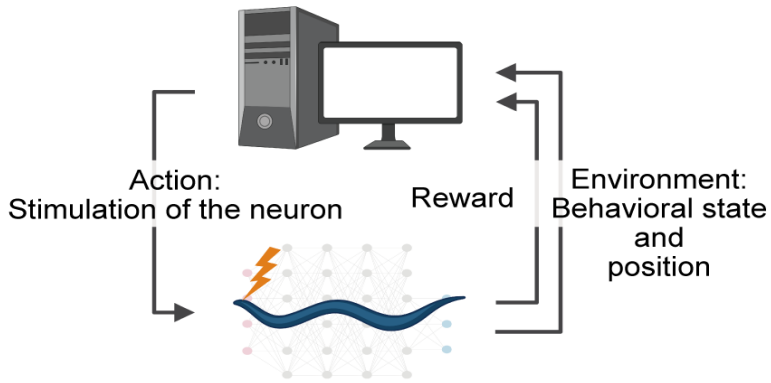


Fig. 2-3 Schematic diagram of reinforcement learning control of *C. elegans*.

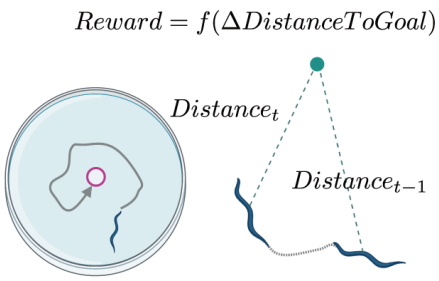


Fig. 2-4 Conceptual diagram of the task of navigation toward the goal point.

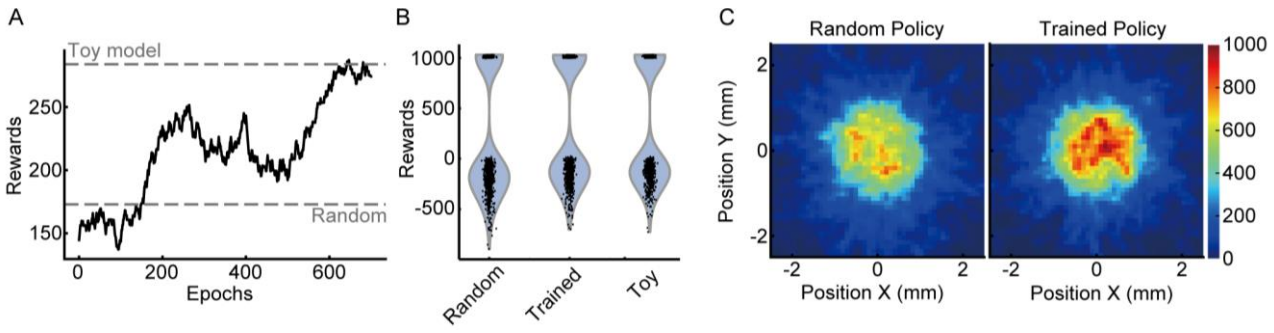


Fig. 2-5 Result of reinforcement learning. Rewards of the trained model in the task using the activation sequence obtained by reinforcement learning, 50% random neuronal activation sequence, and the activation sequence obtained by the toy model. The polygonal line graph shows a representative reward history obtained during training (mean rewards obtained in 300 epochs). (D)-(E) Accomplishment of the task in the test session. (D) Violin plot of reward distribution obtained in the test session of each model. Each dot indicates a reward obtained in an epoch. The rewards obtained by the RL model were significantly larger than those obtained by random sequence (P <0.001, Brunner-Munzel test). (E) The heat map on the left shows the trajectories of 1000 worms using 50% random neuronal activation sequence. The heat map on the right shows the trajectories of 1000 worms using neuronal activation sequence obtained by reinforcement learning. The color bar shows the number of worms/square mm in the simulation space. The goal area is on the center (0–0.2 mm from the center) of the figure and the start points are around the goal area (0.2–1.0 mm from the center).

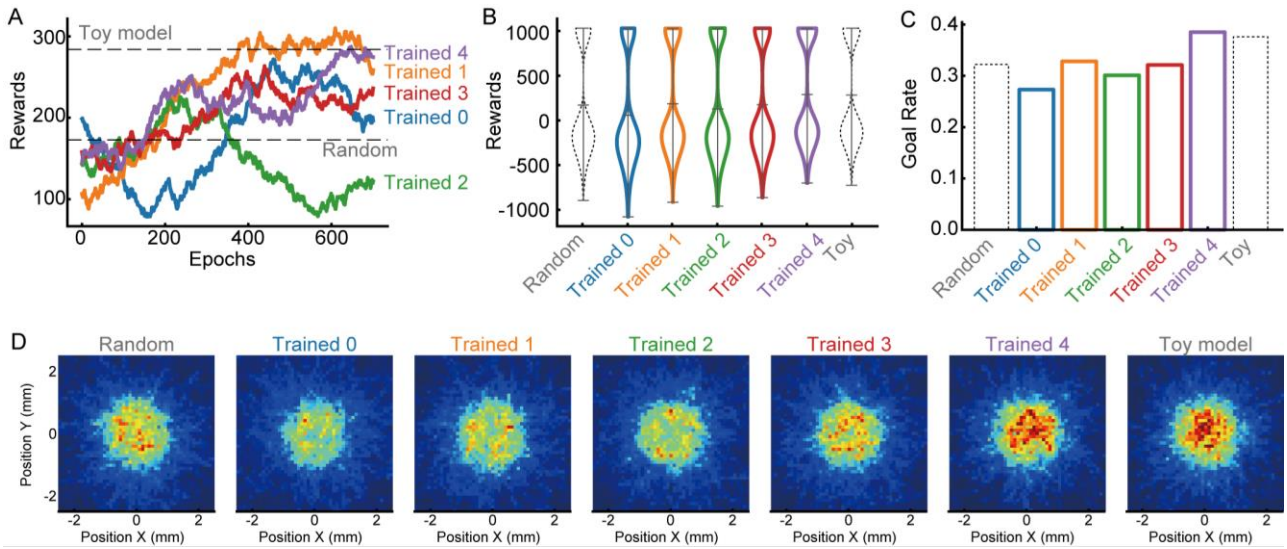


Fig. 2-6 Training result of navigation task. (A) Rewards of the trained model in the task using the activation sequence obtained by reinforcement learning of five different seeds, 50% random neuronal activation sequence, and activation sequence obtained by the toy model. Each polygonal line shows one of five rewards history obtained during training designating mean rewards obtained in 300 epochs. (B)–(D) Accomplishment of the task in the test session. (B) Violin plot of reward distribution obtained in the test session of each model. *P* values of the statistical test between each trained model (seed 0, 1, 2, 3, 4) and random model were $P < 0.001$, $P = 0.680$, $P = 0.038$, $P = 0.725$, and $P < 0.001$ (Brunner-Munzel test). (C) Bar graph indicates mean goal rates of virtual worms in the test session of each model. *P* values of the statistical test between each of trained model (seed 0, 1, 2, 3, 4) and random model were $P = 0.0189$, $P = 0.811$, $P = 0.334$, $P = 1.0$ and $P = 3.73 \times 10^{-3}$ (Chi-square test). (D) The heat maps show the trajectories of 1000 worms using neuron activation sequence obtained by each model. The goal area is located at the center (0–0.2 mm) of the figure, and the start points are located around the goal area (0.2–1.0 mm).

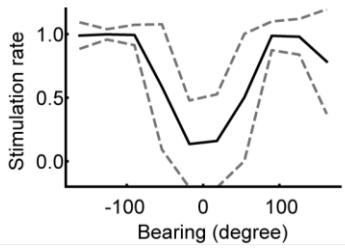


Fig. 2-7 Learned policy of the trained model in the task. The line graph shows the relationship between navigation toward the goal point relative to the virtual worm and neuronal stimulation rate using the neuronal activation sequence obtained by reinforcement learning. The dotted lines indicate the standard deviation.

Appendix

Disentangling animal behavior via temporal conditional-subspace VAE

3.1 Abbreviation

CS-VAE	Conditional subspace - variational autoencoder
VAE	Variational autoencoder
TCS-VAE	temporal conditional subspace variational autoencoder
TCN	temporal convolutional network

3.2 Introduction

In analyzing time series data, it is important to describe the characteristics of the time series data of the target group compared to other groups. Extracting characteristic time series patterns in the time series data of interest is one of the main objectives of time series analysis. The analysis of time series data handled in biological research is no exception. Behavioral data and neural activity data is one of the best examples of this.

With behavioral data in hand, animal ethologists have previously described class-specific behavioral patterns in the animals which they focused on. Many animals show specific types of behavior which depends on their classes such as species, gender, and genotypes.

Determining how class-specific behaviors arise from the nervous system is one of the major goals of neuroscience. In order to achieve this, it is important to elucidate class-specific neural activity patterns in the large-scale data that have been observed. From the perspective of psychiatric disease research, the division into the class of presence or absence of disease can reveal disease-specific patterns of neural activity and behavior.

In this sense, the method of unsupervised extraction of class-specific time series patterns from time series data is very important and is useful in various fields. In previous studies, researchers set the feature of the interest in advance and investigated whether the features change in a class-dependent manner. However, in addition to the large bias of the observer's prior hypothesis, this method was not suitable for the analysis of recent large-scale data. With the development of observational technology, the data handled in biological research has also become multidimensional and larger in size.

To deal with the problem, I propose a method for extracting time series patterns characteristic to a class from this multidimensional, large-scale data by applying machine learning techniques in this chapter.

In this study, I approached the above problem by applying a deep generative model. Deep generative learning is a type of deep learning which tries to reveal the process of data generation. It tries to model the latent stochastic process of the generation of the data that I am interested in. I further applied the deep generation model and thought that I could solve the problem by separating the processes that depend on the

specified class and those that do not depend on the specified class when modeling the process of data generation. I confirmed this hypothesis by applying variational autoencoder (VAE)[99] and its application, conditional subspace VAE (CSVAE)[100], to the data.

VAE is one of the successful methods for deep generative models. Autoencoders constitute a well-known subcategory on the framework which aims at uncovering a projection of high-dimensional input data onto a low-dimensional manifold and to subsequently predict out- put data based on this projection.

There are several ways to handle time series data in a neural network. The most common methods are recurrent neural network and time convolutional network. RNN keeps the history information in the network, while temporal convolutional network (TCN) performs convolutional computation in the time direction. These methods can also be combined with VAE, and several studies have reported [55,101,102]. This study adopts TCN as a method to handle time series data from the viewpoint of interpretability. Temporal convolutional networks (TCN) has ability to compress information in the time direction[103]. Furthermore, some reports show that TCNs outperform RNNs in predicting time series [103].

In this study, I developed a method to extract class-specific patterns from animal behavior data by using the property that CSVAE classifies data into class-specific and class-dependent data. This study dealt with animal behavioral data and neural activity data, but essentially any other time series data is acceptable.

3.3 Result

3.3.1 Development of TCS VAE

First of all, temporal convolutional block (Fig. 3-1) was added to the input part of CSVAE in order to be able to handle time series data. There are two ways to represent time-series data: using RNN patterns and using temporal convolutional blocks. temporal convolutional blocks were chosen for their interpretability and also because there are some research results that show temporal convolutional block performs better than RNN in time series prediction tasks.

Time series data such as behaviors show different features at multiple time levels (msec~min). Therefore, temporal convolutional block can cope with multi-scaling by performing hierarchical handling in the time direction.

A comparison of similar neural networks, autoencoder, temporal autoencoder, conditional subspace VAE, and temporal conditional subspace autoencoder, is shown in Fig. 3-2. The TCSVAE proposed in this research has two latent spaces. Z subspace is a subspace used to handle time series data chunk with common dynamics among classes. The latent variables in Z subspace are trained by adversarial learning in a way that suppresses class-specific information. On the other hand, the W subspace is a subspace that is used when dealing with time series data chunks that have different dynamics among classes.

3.3.2 Toy Data

Next, TCS-VAE was tested on a toy dataset to see if the TCSVAE method of separating class-specific time series patterns and class-independent time series patterns by two VAEs. The toy data set was created as follows. The class 1 group is a sine wave with Gaussian noise on it. The class 2 group is based on the sine wave of class 1, and contains different patterns of high frequency sine waves in the ratio of about 10%. This represents a time series data set in which the base time series pattern is common among the classes, but some of the dynamics are different.

When I trained on this dataset, it was first shown that both classes were able to reconstruct the data (Fig. 3-3A). In particular, in the class 2 group, the characteristic waveforms are also recovered, indicating that not only the network that handles the common parts but also the network that transmits information about the class-specific parts are functioning.

Next, in addition to confirming that the Z and W subspace represent class-specific information and common dynamics among classes, I also confirmed the trajectory in each subspace (Fig. 3-3B). As a result, it was confirmed that in the Z subspace, only the common dynamics between class 1 and class 2 were extracted. On the other hand, in the W subspace, it was confirmed that the topology of the trajectory was different from that of class 1 in the case of characteristic time series patterns.

This result shows that TCSVAE can successfully represent the class-specific dynamics and the common dynamics among classes in different latent spaces.

3.4 Discussion

In this study, it was shown that class-specific behavioral dynamics can be extracted from time series data using a model that combines adversarial learning and deep generative learning. In the future, I would like to apply this model to animal behavioral time series data based on these basic results.

It is useful in many situations to compare the generation process of animal behavior. The classes in this study can be applied to various cases, such as wild type and mutant, male and female, with and without disease, and so on. In the future, I would like to apply them to each pattern and show their effectiveness.

In addition, I would like to investigate how much class characteristics exist in time series data based on the difference in information represented by Z subspace and W subspace. For this purpose, I can consider a model that is trained in two steps as shown in Fig. 3-4 First, I train only VAEs that handle common information among classes, and then I train networks that encode class-typical behaviors in step 2. Then, by comparing the reconstruction rate in step 1 with that in step 2, I can quantify how much class-specific information is included in the input time series data. I would like to develop such a quantitative analysis method in the future.

3.5 Figures

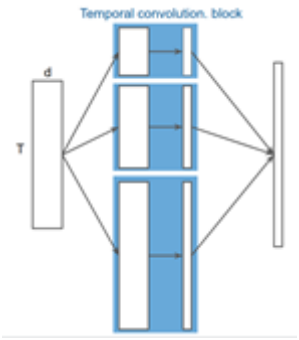


Fig. 3-1 Schematic illustration of temporal convolution block.

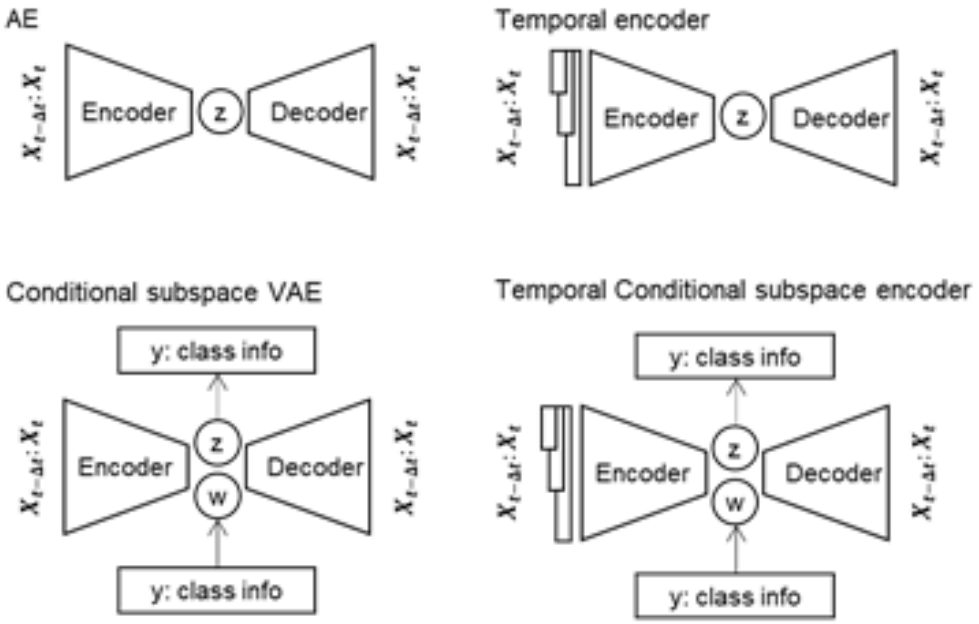
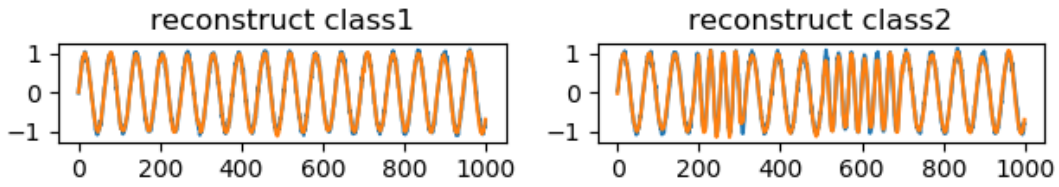


Fig. 3-2 Comparison of related autoencoder-type models.

A



B

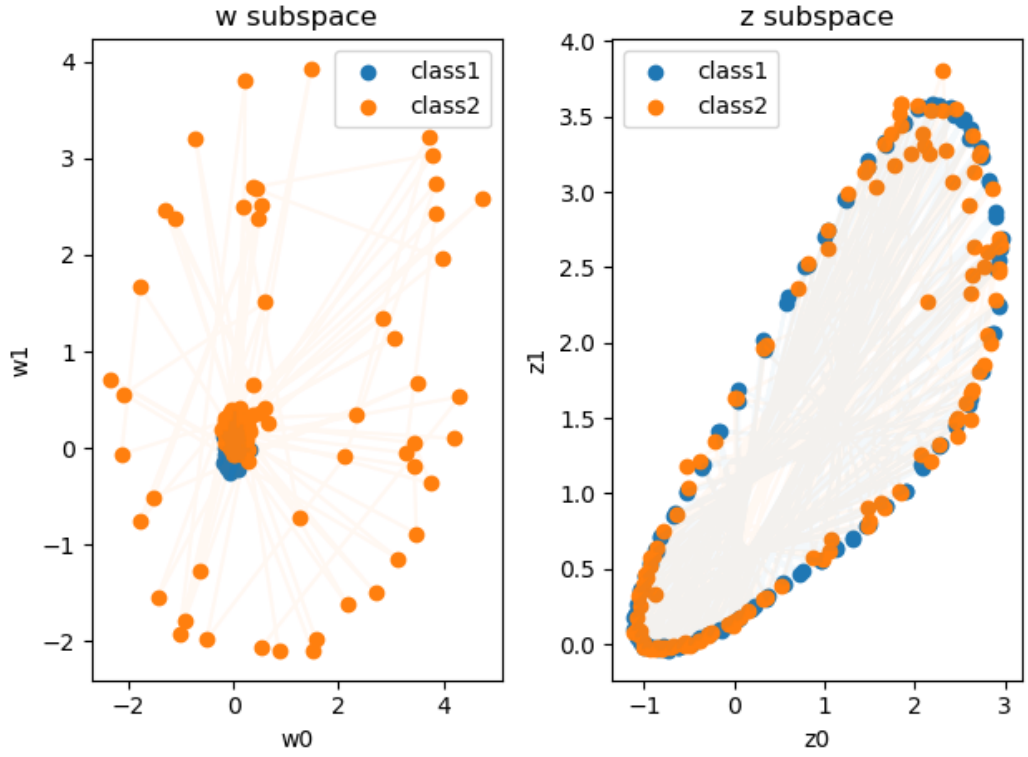


Fig.3-3 Training results of toy model. (A) Reconstruction succeeded in both class 1 and class 2. (B) W subspace handles the class specific information whereas Z subspace handles class independent information.

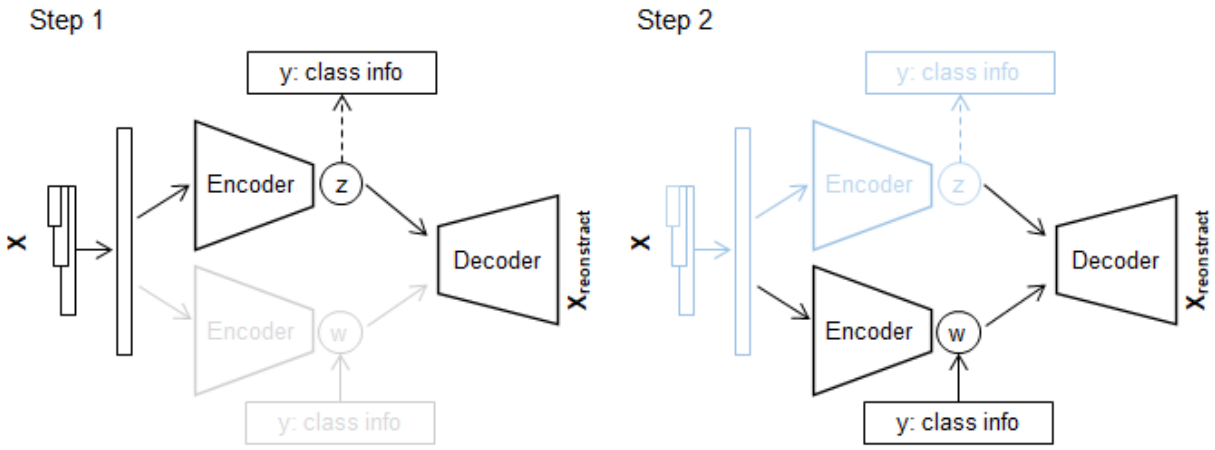


Fig. 3-4 Quantification of class specific information and class independent information.

Conclusion

Three key problems in computational ethology were approached in this dissertation.

In Chapter 1, I constructed a virtual animal using a deep generative model and showed that MDN~RNN can reproduce stochastic animal behavior. The results showed that the MDN~RNN can reproduce stochastic animal behavior. In addition, although it has been known that RNN-based behavioral simulations fall into a motion less state, this problem can be solved by using MDN as a probability distribution for the output. Furthermore, it was shown that behaviors with different dynamics were represented and learned in a disentangled manner depending on the component of the MDN. In the future, our group would like to further develop this model and build a model that matches the actual connectivity pattern of the nervous system.

In Chapter 2, I developed a computational control mechanism for behavior, and showed that a machine can extract behavioral strategies found in nature by automatically searching for a control mechanism that suits the desired task. In order to achieve this, I applied reinforcement learning as a control algorithm, and aimed to replace some of the computational mechanisms of the nervous system with a computer in order to accomplish the tasks that animals perform in nature. It was shown that the computer can acquire and reproduce behavioral strategies similar to those actually performed by animals without prior information.

In Appendix A, I aimed to analyze the topology of the dynamics behind the behaviors of animals belonging to different classes by separating the behaviors that are characteristic of each class from the behaviors that are common regardless of the class. For example, when the behaviors of animals modeled for psychiatric disorders and wild-type animals are acquired, it is important to extract the behaviors exhibited only by the disease model animals and investigate the generation mechanism of the behaviors in order to clarify the diseases. To achieve this goal, I applied the conditional subspace - variational autoencoder (CS-VAE), which takes behavioral data consisting of multiple groups as input and divides them into elements characteristic of the group to which they belong and elements common to all groups in the middle layer of the VAE. The CS - VAE takes behavioral data consisting of multiple groups as input and divides it into elements characteristic of the group to which it belongs and elements common to all groups in the middle layer of the

VAE. This is achieved by minimizing the amount of mutual information between the labels of the groups to which they belong and the features of the latent space in the middle layer. The effectiveness of this method has been verified using toy models, and will be verified using animal behavior in the future.

Each of these three issues has value in the field of computational ethology. With the availability of behavioral measurements, the field of ethology can now cover behavioral data not only in a controlled laboratory setting, but also during free behavior. Therefore, the current challenge is to extract the structures that govern behavior from the obtained time series of behavioral data. Chapter 1 and the appendix present a solution to this problem. In Chapter 2, we proposed a method for machines to mimic animal behavioral strategies using reinforcement learning techniques. Although these methods have been tested on *Caenorhabditis elegans*, I would like to show that these methods can be used to analyze the behavior of various animal species in the future.

Original papers

1) Keita Mori, Haoyu Wang, Naohiro Yamauchi, Yu Toyoshima and Yuichi Iino “Disentangling behavioral dynamics with MDN-RNN” *Proceeding of NeurIPS LMRL worksop* (2020)

2) Keita Mori, Naohiro Yamauchi , Haoyu Wang, Ken Sato, Yu Toyoshima and Yuichi Iino “Probabilistic generative modeling and reinforcement learning extract the intrinsic features of animal behavior” *Neural Networks*, 2022 Jan;145:107-120. doi: 10.1016/j.neunet.2021.10.002.

Acknowledgement

First and foremost, I would like to express my deepest appreciation to my supervisor Dr. Yuichi Iino for his guidance and mentorship throughout the years. Thank you for giving me the opportunity to conduct this study. I would also like to thank Prof. Atsu Aiba who mentored me since undergraduate students. Furthermore, I am grateful for invaluable experience and advice that I received from Prof. Ramdya Pavan.

I am grateful to talented undergrad students Mr. Naohiro Yamauchi and Mr. Haoyu Wang, who conducted the research in Chapter 1 and Chapter 2 together. I also thank Dr. Yu Toyoshima and Mr. Ken Sato, who discussed and developed the experimental paradigm of the research together. Many thanks go to my colleagues who always supported me and discussed the scientific topics.

This research was funded by JSPS KAKENHI and Grant-in-Aid for JSPS Research Fellows. I am also supported by the Graduate Program for Leaders in Life Innovation (GPLLI).

Finally, I would like to express my sincere gratitude to my wife Zhou Jingfang for her warm support and encouragement.

References

1. Marr, D. Vision: A Computational Approach (San Fr. (1982).

2. Nourizonoz, A. *et al.* EthoLoop: automated closed-loop neuroethology in naturalistic environments. *Nat. Methods* 17, 1052–1059 (2020).

3. Mathis, A. *et al.* DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nat. Neurosci.* 21, 1281–1289 (2018).

4. Mathis, A., Schneider, S., Lauer, J. & Mathis, M. W. A Primer on Motion Capture with Deep Learning: Principles, Pitfalls, and Perspectives. *Neuron* 108, 44–65 (2020).

5. Pereira, T. D. *et al.* Fast animal pose estimation using deep neural networks. *Nat. Methods* 16, 117–125 (2019).

6. Marr, D. *Vision: A computational investigation into the human representation and processing of visual information*. (MIT press, 2010).

7. Tinbergen, N. On aims and methods of Ethology. *Z. Tierpsychol.* 20, 410–433 (2010).

8. Berridge, K. C., Fentress, J. C. & Parr, H. Natural syntax rules control action sequence of rats. *Behav. Brain Res.* 23, 59–68 (1987).

9. Dawkins, R. Hierarchical organisation: A candidate principle for ethology. *Growing points in ethology* 7, 54 (1976).

10. Fentress, J. C. & Stilwell, F. P. Letter: Grammar of a movement sequence in inbred mice. *Nature* 244, 52–53 (1973).

11. Burgos-Artizzu, X. P., Dollár, P., Lin, D., Anderson, D. J. & Perona, P. Social behavior recognition in continuous video. in *2012 IEEE Conference on Computer Vision and Pattern Recognition* 1322–1329 (2012).

12. de Chaumont, F. *et al.* Real-time analysis of the behaviour of groups of mice via a depth-sensing camera and machine learning. *Nat Biomed Eng* 3, 930–942 (2019).

13. Giancardo, L. *et al.* Automatic visual tracking and social behaviour analysis with multiple mice. *PLoS One* 8, e74557 (2013).

14. Hong, W. *et al.* Automated measurement of mouse social behaviors using depth sensing, video tracking, and machine learning. *Proc. Natl. Acad. Sci. U. S. A.* 112, E5351-60 (2015).

15. Jhuang, H. *et al.* Automated home-cage behavioural phenotyping of mice. *Nat. Commun.* 1, 68 (2010).

16. Kabra, M., Robie, A. A., Rivera-Alba, M., Branson, S. & Branson, K. JAABA: interactive machine learning for automatic annotation of animal behavior. *Nat. Methods* 10, 64–67 (2013).

17. Lorbach, M., Poppe, R., van Dam, E. A., Noldus, L. P. J. J. & Veltkamp, R. C. Automated Recognition of Social Behavior in Rats: The Role of Feature Quality. in *Image Analysis and Processing — ICIAP 2015* 565–574 (Springer International Publishing, 2015).

18. Rousseau, J. B., Van Lochem, P. B., Gispen, W. H. & Spruijt, B. M. Classification of rat behavior with an image-processing method and a neural network. *Behav. Res. Methods Instrum. Comput.* 32, 63–71 (2000).

19. van Dam, E. A. *et al.* An automated system for the recognition of various specific rat behaviours. *J. Neurosci. Methods* 218, 214–224 (2013).

20. Levitis, D. A., Lidicker, W. Z. & Freund, G. Behavioural biologists don't agree on what constitutes behaviour. *Anim. Behav.* 78, 103–110 (2009).

21. Szigeti, B., Stone, T. & Webb, B. Inconsistencies in C. elegans behavioural annotation. *Cold Spring Harbor Laboratory* 066787 (2016) doi:10.1101/066787.

22. Wahlsten, D. *et al.* Different data from different labs: lessons from studies of gene-environment interaction. *J. Neurobiol.* 54, 283–311 (2003).

23. Berman, G. J., Choi, D. M., Bialek, W. & Shaevitz, J. W. Mapping the stereotyped behaviour of freely moving fruit flies. *J. R. Soc. Interface* 11, (2014).

24. Liu, M., Sharma, A. K., Shaevitz, J. W. & Leifer, A. M. Temporal processing and context dependency in Caenorhabditis elegans response to mechanosensation. *eLife* vol. 7 (2018).

25. Marques, J. C., Lackner, S., Félix, R. & Orger, M. B. Structure of the Zebrafish Locomotor Repertoire Revealed with Unsupervised Behavioral Clustering. *Curr. Biol.* 28, 181-195.e5 (2018).

26. Goupillaud, P., Grossmann, A. & Morlet, J. Cycle-octave and related transforms in seismic signal analysis. *Geoexploration* 23, 85–102 (1984).

27. van der Maaten, L. & Hinton, G. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* 9, 2579–2605 (2008).

28. Cande, J. *et al.* Optogenetic dissection of descending behavioral control in Drosophila. *Elife* 7, (2018).

29. Meng, X., Lee, K. K. & Xu, Y. Human Driving Behavior Recognition Based on Hidden Markov Models. in *2006 IEEE International Conference on Robotics and Biomimetics* 274–279 (ieeexplore.ieee.org, 2006).

30. Nguyen, N. T., Phung, D. Q., Venkatesh, S. & Bui, H. Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model. in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* vol. 2 955–960 vol. 2 (ieeexplore.ieee.org, 2005).

31. Bishop, C. M. *Pattern recognition and machine learning*. (Springer, 2006).

32. Wiltschko, A. B. *et al.* Mapping Sub-Second Structure in Mouse Behavior. *Neuron* 88, 1121–1135 (2015).

33. Wiltschko, A. B. *et al.* Revealing the structure of pharmacobehavioral space through motion sequencing. *Nat. Neurosci.* 23, 1433–1443 (2020).

34. Ackerson, G. & Fu, K. On state estimation in switching environments. *IEEE Trans. Automat. Contr.* 15, 10–17 (1970).

35. Ainsworth, S., Foti, N., Lee, A. K. C. & Fox, E. Interpretable VAEs for nonlinear group factor analysis. *arXiv [cs.LG]* (2018).

36. Chang, C. B. & Athans, M. State Estimation for Discrete Systems with Switching Parameters. *IEEE Trans. Aerosp. Electron. Syst.* AES-14, 418–425 (1978).

37. Fox, E. B., Sudderth, E. B., Jordan, M. I. & Willsky, A. S. Nonparametric Bayesian Learning of Switching Linear Dynamical Systems. *IFAC Proceedings Volumes* 42, 1591 (2009).

38. Ghahramani, Z., Hinton, G. E. & Others. *The EM algorithm for mixtures of factor analyzers*. http://www.cs.utoronto.ca/~hinton/absps/tr-96-1.pdf (1996).

39. Hamilton, J. D. Analysis of time series subject to changes in regime. *J. Econom.* 45, 39–70 (1990).

40. Murphy, K. P. Switching kalman filters. (1998).

41. Linderman, S. W. *et al.* Recurrent switching linear dynamical systems. *arXiv [stat.ML]* (2016).

42. Linderman, S., Johnson, M. & Miller, A. Bayesian learning and inference in recurrent switching linear dynamical systems. *Artif. Intell.* (2017).

43. Glaser, J. I., Whiteway, M., Cunningham, J. P., Paninski, L. & Linderman, S. W. Recurrent switching dynamical systems models for multiple interacting neural populations. *bioRxiv* (2020) doi:10.1101/2020.10.21.349282.

44. Linderman, S., Nichols, A., Blei, D., Zimmer, M. & Paninski, L. Hierarchical recurrent state space models reveal discrete and continuous dynamics of neural activity in C. elegans. *bioRxiv* (2019).

45. Zoltowski, D. & Pillow, J. A general recurrent state space framework for modeling neural dynamics during decision-making. *Conference on Machine …* (2020).

46. Takens, F. Detecting strange attractors in turbulence. in *Dynamical Systems and Turbulence, Warwick 1980* 366–381 (Springer Berlin Heidelberg, 1981).

47. Ahamed, T., Costa, A. C. & Stephens, G. J. Capturing the continuous complexity of behaviour in Caenorhabditis elegans. *Nat. Phys.* (2020) doi:10.1038/s41567-020-01036-8.

48. Tran, Q. H. & Hasegawa, Y. Topological time-series analysis with delay-variant embedding. *Phys Rev E* 99, 032209 (2019).

49. Liu, Z. *et al.* Towards natural and accurate future motion prediction of humans and animals. in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2019). doi:10.1109/cvpr.2019.01024.

50. Ionescu, C., Papava, D., Olaru, V. & Sminchisescu, C. Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *IEEE Trans. Pattern Anal. Mach. Intell.* 36, 1325–1339 (2014).

51. Bütepage, J. & Kragic, D. Human-Robot Collaboration: From Psychology to Social Robotics. *arXiv [cs.RO]* (2017).

52. Gui, L.-Y. *et al.* Teaching robots to predict human motion. in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2018). doi:10.1109/iros.2018.8594452.

53. Tang, Y., Ma, L., Liu, W. & Zheng, W. Long-Term Human Motion Prediction by Modeling Motion Context and Enhancing Motion Dynamic. *arXiv [cs.CV]* (2018).

54. Holden, D., Saito, J., Komura, T. & Joyce, T. Learning motion manifolds with convolutional autoencoders. in *SIGGRAPH Asia 2015 Technical Briefs* (ACM, 2015). doi:10.1145/2820903.2820918.

55. Bütepage, J., Black, M. J., Kragic, D. & Kjellstrom, H. Deep representation learning for human motion prediction and classification. in *Proceedings of the IEEE conference on computer vision and pattern recognition* 6158–6166 (2017).

56. Graving, J. M. & Couzin, I. D. VAE-SNE: a deep generative model for simultaneous dimensionality reduction and clustering. *BioRxiv* (2020).

57. Batty, E., Whiteway, M. R., Saxena, S., Biderman, D. & Abe, T. BehaveNet: nonlinear embedding and Bayesian neural decoding of behavioral videos. https://papers.nips.cc/paper/2019/file/a10463df69e52e78372b724471434ec9-Paper.pdf.

58. Krakauer, J. W., Ghazanfar, A. A., Gomez-Marin, A., MacIver, M. A. & Poeppel, D. Neuroscience Needs Behavior: Correcting a Reductionist Bias. *Neuron* 93, 480–490 (2017).

59. Niv, Y. The primacy of behavioral research for understanding the brain. (2020) doi:10.31234/osf.io/y8mxe.

60. Bin Peng, X. *et al.* Learning agile robotic locomotion skills by imitating animals. in *Robotics: Science and Systems XVI* (Robotics: Science and Systems Foundation, 2020). doi:10.15607/rss.2020.xvi.064.

61. Ijspeert, A. J., Crespi, A., Ryczko, D. & Cabelguen, J.-M. From swimming to walking with a salamander robot driven by a spinal cord model. *Science* 315, 1416–1420 (2007).

62. Ijspeert, A. J. Central pattern generators for locomotion control in animals and robots: a review. *Neural Netw.* 21, 642–653 (2008).

63. Kitano, H. Computational systems biology. *Nature* 420, 206–210 (2002).

64. Sharma, A., Johnson, R., Engert, F. & Linderman, S. Point process latent variable models of larval zebrafish behavior. in *Advances in Neural Information Processing Systems* (eds. Bengio, S. et al.) vol. 31 10919–10930 (Curran Associates, Inc., 2018).

65. Pereira, T. D., Shaevitz, J. W. & Murthy, M. Quantifying behavior to understand the brain. *Nat. Neurosci.* 1–13 (2020).

66. Bishop, C. M. Mixture density networks. (1994).

67. Graves, A. Generating Sequences With Recurrent Neural Networks. *arXiv [cs.NE]* (2013).

68. Ha, D. & Eck, D. A neural representation of sketch drawings. *arXiv [cs.NE]* (2017).

69. Wang, X., Takaki, S. & Yamagishi, J. An autoregressive recurrent mixture density network for parametric speech synthesis. in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 4895–4899 (2017).

70. Martin, C. P. & Torresen, J. RoboJam: A Musical Mixture Density Network for Collaborative Touchscreen Interaction. in *Computational Intelligence in Music, Sound, Art and Design* 161–176 (Springer International Publishing, 2018).

71. Ha, D. & Schmidhuber, J. World Models. *arXiv [cs.LG]* (2018).

72. Zhao, Y., Yang, R., Chevalier, G., Shah, R. & Romijnders, R. Applying Deep Bidirectional LSTM and Mixture Density Network for Basketball Trajectory Prediction. *arXiv [cs.AI]* (2017).

73. Stephens, G. J., Johnson-Kerner, B., Bialek, W. & Ryu, W. S. Dimensionality and dynamics in the behavior of C. elegans. *PLoS Comput. Biol.* 4, e1000028 (2008).

74. Yemini, E., Jucikas, T., Grundy, L. J., Brown, A. E. X. & Schafer, W. R. A database of Caenorhabditis elegans behavioral phenotypes. *Nat. Methods* 10, 877–879 (2013).

75. Martinez, J., Black, M. J. & Romero, J. On human motion prediction using recurrent neural networks. *arXiv [cs.CV]* (2017).

76. Guo, X. & Choi, J. Human motion prediction via learning local structure representations and temporal dependencies. *Proc. Conf. AAAI Artif. Intell.* 33, 2580–2587 (2019).

77. Ellefsen, K. O., Martin, C. P. & Torresen, J. How do mixture density RNNs predict the future? *arXiv [cs.LG]* (2019).

78. Yoshida, K. *et al.* Odour concentration-dependent olfactory preference change in C. elegans. *Nat. Commun.* 3, 739 (2012).

79. Schindelin, J. *et al.* Fiji: an open-source platform for biological-image analysis. *Nat. Methods* 9, 676–682 (2012).

80. Poličar, P. G., Stražar, M. & Zupan, B. openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding. *Cold Spring Harbor Laboratory* 731877 (2019) doi:10.1101/731877.

81. Hyvärinen, A. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Trans. Neural Netw.* 10, 626–634 (1999).

82. Paszke, A. *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library. in *Advances in Neural Information Processing Systems 32* (eds. Wallach, H. et al.) 8026–8037 (Curran Associates, Inc., 2019).

83. Liu, L. *et al.* On the Variance of the Adaptive Learning Rate and Beyond. *arXiv [cs.LG]* (2019).

84. Chatzigeorgiou, M., Bang, S., Hwang, S. W. & Schafer, W. R. tmc-1 encodes a sodium-sensitive channel required for salt chemosensation in C. elegans. *Nature* 494, 95–99 (2013).

85. Foster, D. *Generative Deep Learning*. (O'Reilly Media, Inc., 2020).

86. Sutton, R. S. & Barto, A. G. *Reinforcement Learning, second edition: An Introduction*. (MIT Press, 2018).

87. Niv, Y. Reinforcement learning in the brain. *J. Math. Psychol.* 53, 139–154 (2009).

88. Schultz, W., Dayan, P. & Montague, P. R. A neural substrate of prediction and reward. *Science* 275, 1593–1599 (1997).

89. Mnih, V. *et al.* Human-level control through deep reinforcement learning. *Nature* 518, 529–533 (2015).

90. Glorot, X., Bordes, A. & Bengio, Y. Deep Sparse Rectifier Neural Networks. in (eds. Gordon, G., Dunson, D. & Dudík, M.) vol. 15 315–323 (JMLR Workshop and Conference Proceedings, 2011).

91. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. *arXiv [cs.LG]* (2014).

92. Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12, 2825–2830 (2011).

93. Mnih, V. *et al.* Playing Atari with Deep Reinforcement Learning. *arXiv [cs.LG]* (2013).

94. Kunitomo, H. *et al.* Concentration memory-dependent synaptic plasticity of a taste circuit regulates salt concentration chemotaxis in Caenorhabditis elegans. *Nat. Commun.* 4, 2210 (2013).

95. Berg, H. C. & Brown, D. A. Chemotaxis in Escherichia coli analysed by three-dimensional tracking. *Nature* 239, 500–504 (1972).

96. Krylov, D., Tachet, R., Laroche, R., Rosenblum, M. & Dylov, D. V. Reinforcement Learning Framework for Deep Brain Stimulation Study. *arXiv [q-bio.NC]* (2020).

97. Pineau, J., Guez, A., Vincent, R., Panuccio, G. & Avoli, M. Treating epilepsy via adaptive neurostimulation: a reinforcement learning approach. *Int. J. Neural Syst.* 19, 227–240 (2009).

98. Rao, R. P. Towards neural co-processors for the brain: combining decoding and encoding in brain-computer interfaces. *Curr. Opin. Neurobiol.* 55, 142–151 (2019).

99. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. *arXiv [stat.ML]* (2013).

100. Klys, J., Snell, J. & Zemel, R. Learning Latent Subspaces in Variational Autoencoders. *arXiv [cs.LG]* (2018).

101. Chung, J. *et al.* A Recurrent Latent Variable Model for Sequential Data. *arXiv [cs.LG]* (2015).

102. Lu, Y., Kumar, K. M., s. Nabavi, S. & Wang, Y. Future Frame Prediction Using Convolutional VRNN for Anomaly Detection. in *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* 1–8 (2019).

103. Bai, S., Zico Kolter, J. & Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv [cs.LG]* (2018).