https://doi.org/10.20965/jaciii.2021.p0335

Paper:

Materializing Architecture for Processing Multimodal Signals for a Humanoid Robot Control System

Motohiro Akikawa and Masayuki Yamamura†

Department of Computer Science, School of Computing, Tokyo Institute of Technology
4259 Nagatsuta-cho, Midori-ku, Yokohama, Kanagawa 226-8503, Japan
E-mail: m_akikawa@ali.c.titech.ac.jp, my@c.titech.ac.jp
†Corresponding author

[Received October 21, 2020; accepted February 25, 2021]

In recent years, many systems have been developed to embed deep learning in robots. Some use multimodal information to achieve higher accuracy. In this paper, we highlight three aspects of such systems: cost, robustness, and system optimization. First, because the optimization of large architectures using real environments is computationally expensive, developing such architectures is difficult. Second, in a real-world environment, noise, such as changes in lighting, is often contained in the input. Thus, the architecture should be robust against noise. Finally, it can be difficult to coordinate a system composed of individually optimized modules; thus, the system is better optimized as one architecture. To address these aspects, a simple and highly robust architecture, namely memorizing and associating converted multimodal signal architecture (MACMSA), is proposed in this study. Verification experiments are conducted, and the potential of the proposed architecture is discussed. The experimental results show that MACMSA diminishes the effects of noise and obtains substantially higher robustness than a simple autoencoder. MACMSA takes us one step closer to building robots that can truly interact with humans.

Keywords: deep learning, robot control, multimodal information, materializing architecture, robust architecture

1. Introduction

The purpose of this paper is to propose a materializing architecture for robot control systems, such as for humanoid robots. With advances in robotics, robots that are capable of interacting with humans have been developed [1], e.g., ASIMO [2] and Pepper [3]. However, these robots interact with humans only under limited conditions. No robot exists that can interact with humans perfectly under all conditions.

Conventional mainstream controlling systems for robots consist of classical programs that have been adopted and extended. However, these systems cannot always adapt to some well-discussed problems, such as the frame problem, nor can they perform complex actions [3].

In recent years, system design using machine learning, such as deep learning, has become mainstream [4–6]. Using deep learning and machine learning techniques, high-accuracy recognition has become possible in computer vision [5–14]. Given the achievements in computer vision, these techniques have been adapted to robot systems as well [15–18]. In addition, there are some systems that use multimodal information, e.g., a combination of images and sounds, as input to obtain higher accuracy [19–21]. However, the studies that use deep learning with multimodal information as input have three significant challenges. The first is the increasing cost of optimizing networks: this includes not only the computational cost but also the cost of designing an architecture. If multimodal information is simply used as input to one neural network, bloated network structures cause computational costs to increase [7, 8, 19]. Although restricted network structures are used on some systems, it is not clear how networks should be restricted [15]. Thus, the challenge of restricting the network structure is faced by the system designer. Further, in designing an architecture consisting of some types of networks, the way the networks connect to each other is most important. Ramachandram and Taylor [6] mentioned that "architecture design has been more an art than a science" because designers need to pay considerable attention to the structure. However, architecture design should be a science, because an architecture cannot be expanded and adapted to other tasks without a good understanding of the intricacies of the structure.

The second challenge is robustness. A system exists that processes time-series data as input by treating several frames of information as one input [20]. However, this system is unstable at the beginning of a motion or until all inputs include environmental changes that should be treated as noise, such as lighting changes.

State-of-the-art methods only use an approximate function of the neural network and generate output based on these features. These methods address the challenge of robustness to achieve high accuracy for specified tasks in experiments. However, for communicating with humans, it is more important to reduce noise in real-world environments and adapt to unknown situations.

The third challenge is a best-effort optimized system.

In some systems, deep learning is used only for object recognition, and the dedicated software for a specific robot is used to control it [21]. If the sensor and drive modules are optimized individually and either module's specifications are changed, the system immediately becomes unstable. It is also well-known that if the sensor modules are optimized but drive modules are not, the robot may not perform the action expected by the programmer. Therefore, optimizing a robotics system as one architecture that includes all modules would be appropriate.

In this research, considering the three aforementioned aspects, which are cost, robustness, and a best-effort optimized system, a memorizing and associating converted multimodal signal architecture (MACMSA) is proposed. The novelty of the proposed architecture lies in its combination of components. Notably, the control system for humanoid robots that combines two completely different types of networks, namely autoencoders and Hopfield networks, is completely novel; autoencoders and Hopfield networks are categorized as feed-forward neural networks and fully interconnected neural networks, respectively. Further, the performance and limitations of the proposed architecture were experimentally investigated. Finally, the validity of MACMSA is discussed with respect to the three aspects studied herein, when the architecture is implemented in robots.

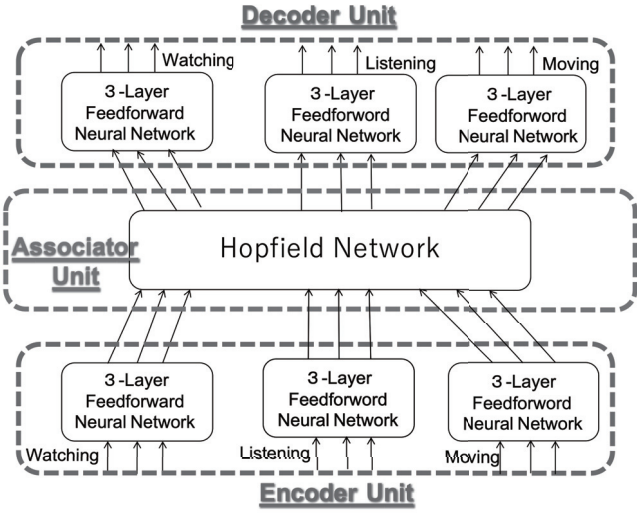

Fig. 1. Architecture is composed of three modules, namely, encoders, decoders, and an associator. Each mode of information is input to each encoder. Then, the encoders output information converted to binary data. The outputs from the encoders are concatenated into one vector to be the input for the associator unit. The vector obtained as the output from the associator is divided to be inputs to the decoders. The system outputs each mode of information from each decoder.

2. Materials and Methods

2.1. Outline of the Architecture

The architecture comprises three modules (**Fig. 1**). 1) Associator: a memory device that behaves as an associative memory and is the most important module. Using a sparse Hopfield network [22] as the associator, the robots store and recall strongly denoised signals. Previous works, [23] and [24], studied the implemented associative memory on robots. 2) Encoders: address translators that convert a real number, which is an input from sensors, to a binary value. The reason the encoders output a binary value is that a Hopfield network obtains a binary value as an input and processes a binary value. 3) Decoders: address translators that convert a binary value, which is an output from a Hopfield network, into a real number. The reason the decoders output a real number is that the inputs for many types of sensors are real numbers. The encoder and the decoder are independently implemented as three-layer feedforward neural networks for each mode of information. In this study, we assume that three modes of information, namely visual information, audio information that has been transformed using Fourier transform, and actuator states (including angle, angular velocity, and angular acceleration) are fused.

2.2. Optimization Procedure

MACMSA is optimized by three steps: 1) the encoder and decoder are optimized; 2) all training data are encoded into binary patterns by the optimized encoders; and 3) the associator stores all the encoded training data. Each step is discussed in detail below.

Step 1 involves optimizing the encoder and decoder. The encoder and decoder of each mode of information are optimized as sparse autoencoders [25]. Many systems that implement autoencoders are available in robotics [19, 20, 26]. Typically, only half of the network structure is used to detect the features of the data or compress the dimensions of data after optimizing the autoencoder [27]. In this study, one half of the network structure is used as the encoder and the other half as the decoder. The output layer of the encoder and the input layer of the decoder share the same layer. Thus, the encoder and decoder can be optimized together as a five-layer autoencoder (**Fig. 2**). All layers use the sigmoid function as the activation function and have the same value of gain, except for the output layer of the encoder. The output of the encoder is required to approximate a binary value because the Hopfield network, which is used as the associator, processes binary values. Backpropagation with stochastic gradient descent optimizes the autoencoder. The cost function used to optimize the sparse autoencoder is given by adding a squared error with a sparse regularization term, i.e., the Kullback–Leibler divergence (KL) [28]:

$$E(w) = \sum_n \|\boldsymbol{y}_n - \boldsymbol{t}_n\|^2 + \beta \sum_j KL(\rho \| \hat{\rho}_j), \quad \ldots \quad (1)$$

where $E(w)$ is the cost function for weights w; n is an index of data; y_n is the n-th output from the autoencoder; t_n is an answer signal for the n-th input; β is the hyperpa-

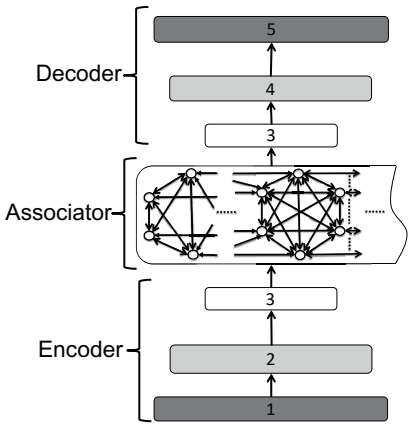

Fig. 2. One autoencoder is divided at Layer 3 to create an encoder and decoder. At the output layer of the encoder, a part of a vector to be input to the associator is obtained. The input layer of the decoder receives a part of the vector as an input.

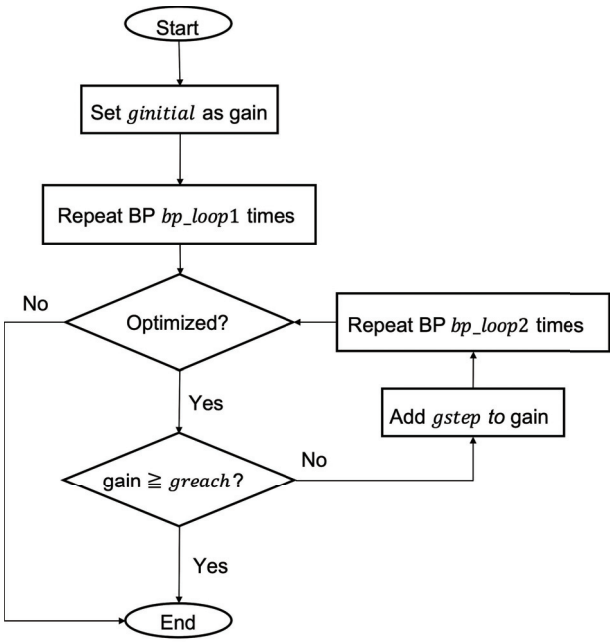

Fig. 3. First, the encoder adopts a low initial gain (*ginitial*) and updates the weight for *bp_loop*1 iterations using back-propagation. Then, the encoder adopts a new gain by adding *gstep* to the old gain and updates the weights for *bp_loop*2 iterations. Thereafter, the encoder increases the gain by *gstep* at every *bp_loop*2 iteration and updates the weights until the gain reaches *greach*.

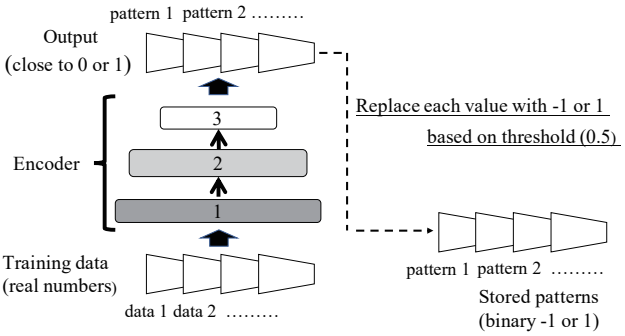

Fig. 4. First, all training data are encoded by the encoder. Then, all outputs are replaced with 0s or 1s using 0.5 as the threshold. In addition, 0 is replaced with -1 for programming convenience. These patterns with -1 or 1 are stored at the associator.

rameter that controls the effect of the sparse regularization term on the cost function; j is the index of neurons composing the network; ρ is the target value for sparseness; and $\hat{\rho}$ is the actual measured value.

As a result, the derivative form for updating the weights is as follows [28]:

$$\frac{\partial E}{\partial u_j^{(l)}} = \left\{ \sum_k \frac{\partial E}{\partial u_k^{(l+1)}} w_{kj}^{(l+1)} + \beta \left(-\frac{\rho}{\hat{\rho}_j} + \frac{1-\rho}{1-\hat{\rho}_j} \right) \right\} f'\left(u_j^{(l)}\right), \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

where j and k are indices of neurons; thus, u_k and u_j denote the errors from the k-th and j-th neurons, respectively; l denotes the index of the layer; and f' is the derivative of the activation function.

As noted previously, the gain at the output layer of the encoder is increased. To increase the value of gain, the optimization and setting of the gain are repeated following the flowchart depicted in **Fig. 3**.

Step 2 involves generating encoded patterns for the associator. The encoders optimized in Step 1 are fed all the training data as input, and they output the encoded patterns. These patterns are approximately binary but are still real numbers at this point. Therefore, following the operation illustrated in **Fig. 4**, all patterns stored by the associator are replaced by binary patterns. The system adopts this replacement even during operation of the system.

Step 3 involves storing the patterns in the associator. Each weight for the Hopfield network is determined as J_{ij} [22]:

$$J_{ij} = \frac{1}{M} \sum_{\mu=1}^{M} (\zeta_i^\mu - a)(\zeta_j^\mu - a), \quad \ldots\ldots (3)$$

where M denotes the number of stored patterns; μ is an index of stored patterns; a is an average pattern; ζ_i is a signal for the i-th neuron and is obtained from the output layers of the encoders; and i and j are indices of neurons for the Hopfield network – this j is different from the j for the autoencoder in Step 1. For the sparse Hopfield network, the recall function is given by Eqs. (4) and (5) [22].

$$x_i^{t+1} = F\left(\sum_{j \neq i}^{N} J_{ij} x_j^t + h \right), \quad \ldots\ldots\ldots (4)$$

$$F(u) = \mathrm{sgn}(u) - b, \quad \ldots\ldots\ldots\ldots (5)$$

Table 1. Number of neurons in each autoencoder and loop values.

Autoencoder structure	Input layer	Hidden layer 1	Hidden layer 2	Hidden layer 3	Output layer	*bp_loop*1	*bp_loop*2
Structure 1	1000	750	500	750	1000	15000	7500
Structure 2	500	375	250	375	250	10000	5000
Structure 3	100	75	50	75	100	7500	3750

where N denotes the number of neurons; x_j^t denotes the state of each neuron at time t; h represents the shift, and the signal is maximized when $h = a(1-a^2)$; and b is the bias – cross-talk noise is minimized when $b = a$. During system operation, the Hopfield network obtains the input as the initial state from the encoders, and outputs a steady-state pattern according to Eq. (4). In addition, the theoretical storage capacity is calculated by Eq. (6) and approximated via a signal-to-noise analysis [22].

$$\alpha_C(a) = \frac{\alpha_C(0)}{(1-a^2)}, \quad \ldots\ldots\ldots\ldots\ldots (6)$$

where $\alpha_C(0) = 0.138N$, and N denotes the number of neurons.

2.3. Experimental Conditions

Three groups of experiments were conducted – Group A: verification of concept, Group B: observation of performance and limitations, and Group C: comparison with a simple autoencoder. Here, experiments in Group A are denoted by Experiment 1-1, 1-2, 1-3, and 1-4 and Group B are similarly denoted by Experiment 2-1, 2-2, and 2-3. Group C consists of one experiment. Group A involved four experiments using pseudodata to show that MACMSA performs in accordance with the concept. In Experiments 1-1–1-3, the performance of each component or a part of the architecture was investigated. In Experiment 1-4, the performance of the entire architecture was verified. Group B involved three experiments in which the sizes of the encoder, associator, and decoder, directly affecting the storage capacity, were varied to clarify the relationship between accuracy and the storage capacity. The three experiments in Group B corresponded to Experiments 1-2–1-4. In Group C, a simple autoencoder that processes three modalities concurrently was used. In summary, Group A conducted basic investigations of the performance of the proposed architecture; Group B performed observations when the parameters (i.e., network size and dimensions of data) were changed based on the results of Group A; Group C conducted experiments with the parameters used in Group A, and was the most appropriate to reveal the differences between the proposed architecture and a simple autoencoder.

2.3.1. Verification of Concept (Group A)

Experiment 1-1. This experiment demonstrates the optimization of the three autoencoders. In this experiment, to demonstrate that the proposed architecture can handle different dimensions of data, the input layer sizes of the largest and the smallest encoders were set to 1,000 and 100, respectively. One function of decoders is to reduce the dimensions. Thus, the size of the hidden layers may be smaller than that of the input layers. However, the sum of neurons in all Hidden layer 3s is the number of neurons in the Hopfield network, which directly affects the storage capacity of the Hopfield network. Therefore, to avoid considering the storage capacity in this experiment, the size of the Hidden layer 3s should be sufficiently large. For these reasons, the number of neurons in the Hidden layer 3s was half that of the input layers, and the number of neurons in Hidden layer 2s was three quarters of that of the input layers. The input layer size of the mid-sized network was set to 500, which is almost intermediate in size. To confirm that the encoders and decoders can process data with various averages and variances, the training data were chosen as pseudodata composed of a combination of three averages (0.25, 0.5, 0.75) and three standard deviations (0.25, 0.28, 0.31). Thus, 729 combinations are possible, all of which were tested. The three averages and three standard deviations reproduce the case that each encoder obtains inputs from different types of sensors, which output data with different averages and standard deviations. Ten sets of pseudodata were generated from normally distributed random numbers. The range of possible values was $[0.0, 1.0]$. To avoid considering the storage capacity of the Hopfield network, the amount of pseudodata should be small because the amount of training data determines the number of patterns to be stored. For each combination, 10 trials were conducted with different seeds; thus, 7,920 conditions were tested. **Table 1** lists the number of neurons in the autoencoder. All the weights of the three autoencoders were initialized with normally distributed random numbers with an average of 0.0 and standard deviation of 0.3. *ginitial* was set to 1.0 to make the activation function the standard sigmoid function; *greach* was set to 10 to enable the encoders output approximately binary values. Setting a high *gstep* causes an optimization failure because the network structure is drastically changed. *bp_loop*1 and *bp_loop*2 should be set with α (i.e., learning rate) to properly optimize the autoencoder. The sparseness of patterns for the associator can be controlled by ρ. If β is large, e.g., 3, the autoencoder extracts the input data directly as features. However, if β is very small, e.g., 0.001, the autoencoder expresses all data by one feature [28]. Thus, β should be

Table 2. Hyperparameter settings.

ginitial	*greach*	*gstep*	α	ρ	β
1.0	10.0	0.5	0.001	0.4	0.2

set carefully. *gstep*, *bp_loop*1, *bp_loop*2, α, ρ, and β were fine-tuned through supplementary experiments. **Table 2** shows the hyperparameters used to optimize the three autoencoders.

Experiment 1-2. The three autoencoders optimized in Experiment 1-1 were inputted with noise, and the amount of error in the output was measured. The input consisted of training data with noise levels of 0–50% in increments of 2%. A noise level of 50% indicates that half the dimensions of all training data are changed. It cannot be assumed that half the input data are changed in exactly the same way in a given situation. Thus, in this experiment, the maximum noise level was set to 50%. In this experiment, the way outputs will be changed against tiny variations in inputs needs to be determined. Therefore, the degree of small changes was set as 2%. Noise was added by replacing a portion of the training data with a uniform random number in the range $[-1.0, 1.0]$ for each mode of information. The reason the minimum noise value was -1.0 is that some types of sensors work as analog devices, and they should output values outside the defined range as noise. The probability distribution of data and that of noise are different in real-world environments. Therefore, noises were generated with a uniform random number. If a noise level of 25% is set, all training data are set to baseline, and 25% of all training data of every modality are replaced by uniform random numbers. For instance, for the first modality, 250 dimensions, which constitute 25% of the training data, are chosen at random for each of the training data, and then replaced by uniformly distributed random numbers. Similarly, for the second and third modalities, 125 and 25 dimensions were respectively chosen at random for each of the training data and replaced by uniformly distributed random numbers. When the noise was 0%, the input was the same as the training data. Each combination of the training data and noise rates were tested 10 times. Thus, 72,900 conditions were tested in one noise level and 1,895,400 conditions were tested in total, because there were 26 steps of noise level.

Experiment 1-3. To test robustness, the encoders and the associator were inputted with noise, and the amount of information that the associator can recall accurately was measured. The autoencoders optimized in Experiment 1-1 were used as the encoders. The associator was initialized using the patterns generated by the methods described in Steps 2 and 3 in Section 2.2. The encoders were inputted with noise levels of 0–50% in intervals of 2%, similar to that in Experiment 1-2. This test was also performed 10 times for all combinations and with all noise rates. Thus, 1,895,400 conditions were tested in total.

Experiment 1-4. To test robustness against noise, in this experiment, the entire system was tested. The encoders optimized in Experiment 1-1 encoded the input with noise, similar to that in other experiments. The associator was inputted the pattern from the encoders, and then recalled the output pattern. The decoders decoded the pattern to the original formatted data. If the system operates properly, even if the input contains noise, the output from the system should be similar to the original training data. Similar to that in the other experiments, the system obtained input with noise levels of 0–50% in increments of 2%, and each noise level and all training data were tested 10 times. Thus, 1,895,400 conditions were tested in total, same as that in other experiments.

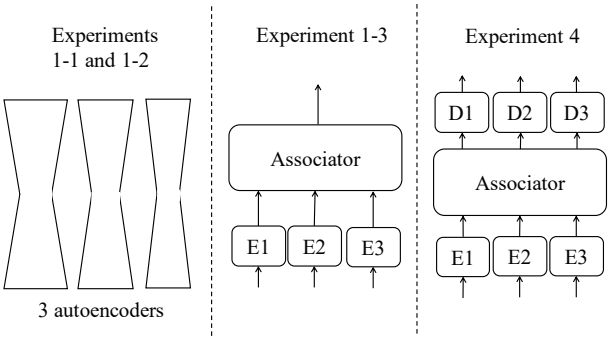

Fig. 5. Experiment 1-1 evaluated the performance of three autoencoders for the optimization of the encoders and decoders. Experiment 1-2 measured the error of the three autoencoders. Experiment 1-3 tested the associator with input from the encoder. Experiment 1-4 measured the error of the system.

Figure 5 shows the networks focused on each experiment and the measurement particulars.

2.3.2. Observation of Performance and Limits (Group B)

In this group, three experiments corresponding to Experiments 1-2–1-4 were conducted. In these experiments, four architecture sizes were tested for a fixed number of 10 training datasets. 10 datasets could be very few for training, but the smallest architecture, which is the minimum size of the proposed architecture, can store only up to 14 non-sparse patterns because of the storage capacity. Experiment 2-1 was conducted during operation with three simple autoencoders. Experiment 2-2 was an experiment on the associator using the encoders and the associator. Experiment 2-3 was an experiment on the entire system. **Table 3** shows the number of neurons for each architecture and the values of *bp_loop*1 and *bp_loop*2. The number of neurons was determined by considering the storage capacity of the Hopfield network and the size of autoencoder that can be optimized. Note that autoencoders smaller than Structure 3 of architecture 1 (Arch 1) could not be optimized by the proposed method. The number of training data was fixed, and the number of neurons for the Hopfield network was decreased to control the loading rate, which is the amount of storage used for patterns with respect to the storage capacity. Three averages (0.25, 0.5, 0.75) and one standard deviation (0.28)

Table 3. Components of each system.

Architecture	Autoencoder structure	Input layer	Hidden layer 1	Hidden layer 2	Hidden layer 3	Output layer	*bp_loop*1	*bp_loop*2
Arch 1	Structure 1	100	75	50	75	100	7500	3750
	Structure 2	70	53	35	53	70	7500	3750
	Structure 3	40	30	20	30	40	7500	3750
Arch 2	Structure 1	180	135	90	135	180	7500	3750
	Structure 2	120	90	60	90	120	7500	3750
	Structure 3	60	45	30	45	60	7500	3750
Arch 3	Structure 1	360	270	180	270	360	7500	3750
	Structure 2	240	180	120	180	240	7500	3750
	Structure 3	120	90	60	90	120	7500	3750
Arch 4	Structure 1	1000	750	500	750	1000	15000	7500
	Structure 2	500	375	250	375	250	10000	5000
	Structure 3	100	75	50	75	100	7500	3750

were used to generate the training pseudodata. Unlike in Group A, a combination of averages was not considered; thus, training data with the same averages were used for optimizing structures 1, 2, and 3. For all average values of the training data, 10 trials were performed with different seeds. Thus, 300 conditions were tested for one noise level and 7,800 conditions were tested in total. The hyperparameters (**Table 2**) and other conditions were the same as those in Group A.

2.3.3. Comparison with a Simple Autoencoder (Group C)

The proposed architecture, which has the same number of neurons as that in Group A (**Table 1**), was compared with a simple autoencoder that processes three modalities concurrently. It was assumed that the three sensors output data with different averages. The averages of Modes 1, 2, and 3 were 0.25, 0.5, and 0.75, respectively; the standard deviation of all modes was 0.28. For the proposed architecture, Encoders 1, 2, and 3 were obtained by Modes 1, 2, and 3, respectively. The number of training datasets was 10. The hyperparameters used to optimize the proposed architecture were the same as those in Group A (**Table 3**). The five-layer autoencoder was obtained by simply concatenating three modal sets of information as inputs (**Fig. 6**). The number of neurons in the input layer for the autoencoder was 1,600, which was the sum of the neurons in the input layers in the encoders of the proposed architecture. Similarly, the number of neurons in each layer was the sum of neurons in each layer of the encoders and the decoders of the proposed architecture. Thus, Layers 2 and 3 contained 1,200 and 800 neurons, respectively. The total number of neurons in the autoencoder was the same as that in the encoders and the decoders of the proposed architecture. The first part of the input layer, containing 1,000 neurons, obtained Mode 1 as input. Similarly, the second and third parts of the input layer, containing 500 and 100 neurons, obtained Mode 2 and Mode 3, respectively. The loss function was simply the sum of squares. Furthermore, the loss function took 75,000 backpropagation iterations with stochastic gradient descent to converge completely. The gain for the sigmoid function was 1.0, to make the activate function the standard sigmoid function. The learning rate α was 0.001, which was the same as that for the proposed architecture. Both the proposed architecture and the autoencoder were tested in 10 trials with different seeds; 10 tests were run for each noise level for both cases. Thus, 2,600 tests were run in total.

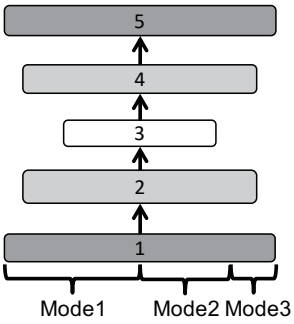

Fig. 6. Simple autoencoder consisting of five layers. The autoencoder obtained three modality inputs. The averages of Modes 1, 2, and 3 were 0.25, 0.5, and 0.75, respectively. The total number of neurons was the same as the total number of neurons of the encoder and the decoder in the proposed architecture.

3. Results

3.1. Verification of Concept (Group A)

Experiment 1-1. The average of the evaluations of all autoencoders was shown to converge (**Fig. 7**). The evaluation value degraded periodically because the network was slightly modified by resetting the gain.

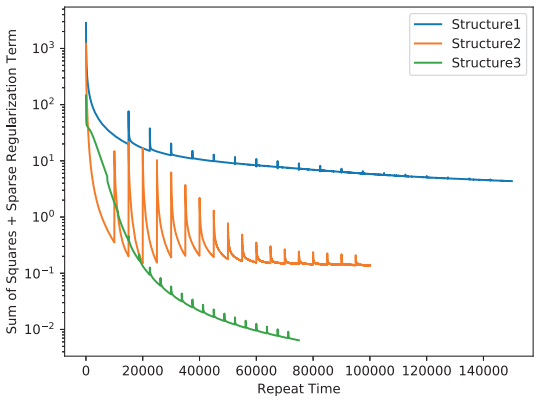

Fig. 7. Result of optimizing each autoencoder. The x-axis represents the number of iterations of backpropagation, and the y-axis represents the values of the sum of squares + the sparse regularization term (Eq. (1)).

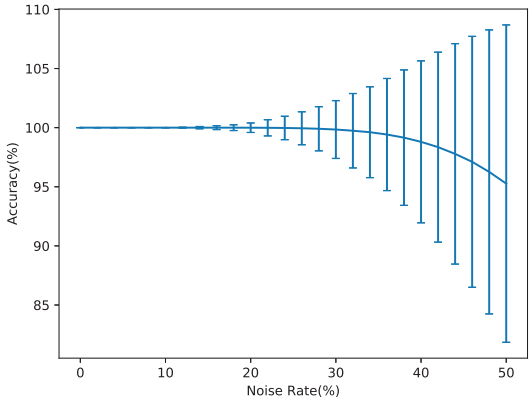

Fig. 9. Percentage of correct numbers when the associator is given input from the encoder that includes noise. The y-axis represents accuracy, measured as the normalized Hamming distance. The error bars indicate standard deviation.

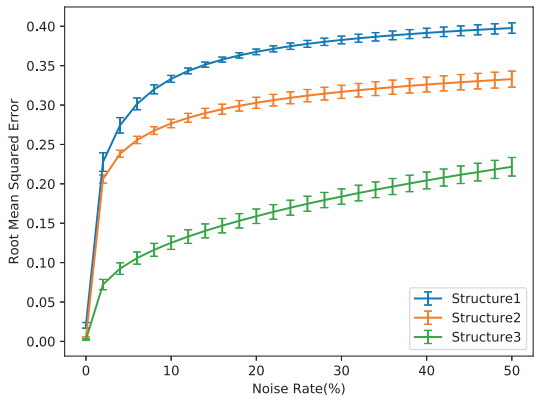

Fig. 8. Error in the output from the autoencoders when the input contains noise. The x-axis represents the noise rate, y-axis represents root mean squared error, and the error bars indicate standard deviation.

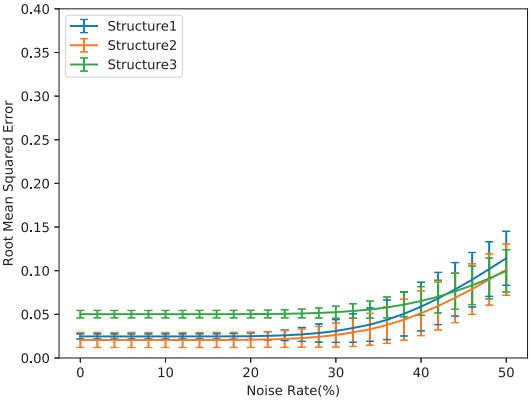

Fig. 10. Error on the output from the system when the input contains noise. The error bars indicate standard deviation.

Experiment 1-2. As the size of the network structure increased, the error at a 0% noise level increased but was still quite low (**Fig. 8**). At a 2% noise level, the errors for Structures 1, 2, and 3 were ∼0.23, ∼0.2, and ∼0.07, respectively. The error increased sharply at a noise level of 2%, considering the low errors at a noise level of 0%. A tendency for the error to increase gradually as the error increased sharply was observed, regardless of the structure size.

Experiment 1-3. When the noise rate was 0–12%, the accuracy was 100%, indicating that the correct pattern was recalled (**Fig. 9**); thereafter, the percentage of correct outputs gradually decreased. As the noise level increased and the percentage of correct outputs decreased, the standard deviation increased.

If the correct pattern was not recalled, an incorrect pattern was recalled, or the system was in an unstable state. When the level of noise was low and the accuracy was not 100%, it is assumed that the input pattern was not sufficiently close to recall the correct pattern. Instead, a pattern that was relatively similar to the correct pattern was recalled, because the input was not sufficiently close to recall the correct pattern. The high accuracy indicates that the Hamming distance between the correct pattern and the output pattern was short. When the level of noise was high, it is assumed that the input pattern was too far from the correct pattern that a totally different pattern was recalled. Depending on the stored patterns, even if the noise rate was high, there were cases in which the correct pattern was recalled. Consequently, when the level of noise was low, the standard deviation was small; when the level of noise was high, the standard deviation increased.

Experiment 1-4. For up to a 30% noise level, the system maintained extremely low levels of error (**Fig. 10**). Beyond the 30% noise level, the errors from the system increased almost linearly. At a 50% noise level, the er-

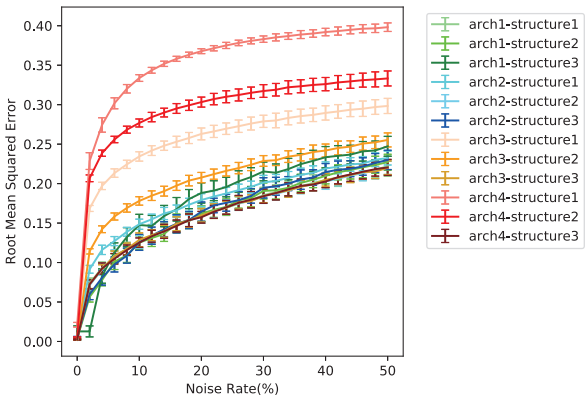

Fig. 11. Error on the output from autoencoders in each architecture when the input contains noise. The error bars indicate standard deviation.

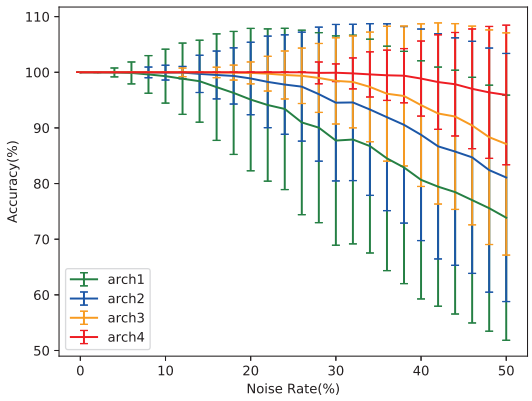

Fig. 12. Percentage of correct outputs from the associator of each architecture. The *y*-axis represents the accuracy, measured using the normalized Hamming distance. The error bars indicate standard deviation.

Table 4. α, storage capacity, and loading rate for each architecture.

Architecture	α	Storage capacity	Loading rate
Arch 1	0.211	0.144N	66.1%
Arch 2	0.206	0.144N	38.6%
Arch 3	0.203	0.144N	19.4%
Arch 4	0.205	0.144N	8.7%

ror from Structure 1, which had the worst accuracy in this experiment, was ∼0.1. In contrast, in Experiment 1-2, the error from Structure 3, which had the best accuracy at 50% noise rate, was larger than 0.2.

The results of Experiment 1-2 and this experiment suggest that including an associator in the system can decrease the error and obtain substantially better robustness than a simple autoencoder.

3.2. Observation of Performance and Limits (Group B)

Experiment 2-1. When the noise level was 0%, the error was quite low (**Fig. 11**); however, as the noise rate increased, the error increased. As in Experiment 1-2, the error exhibited a tendency to initially increase sharply, and then increase gradually as the noise increased. This tendency was observed regardless of the structure size.

Experiment 2-2. **Table 4** lists the measured α, which was the average pattern, theoretical storage capacity of the Hopfield network, and loading rate. On Arch 2, Arch 3, and Arch 4, the correct pattern was recalled even if the input to the encoders contained noise (**Fig. 12**). The ability to recall the correct pattern even with high levels of noise depended on the size of the associator. As the size of the associator increased, this ability improved.

Experiment 2-3. The robustness of the entire system against noise increased as the loading rate decreased (**Fig. 13**). In particular, the entire system of Arch 4, loaded under 10% of the storage capacity, maintained extremely low error even if the inputs contained up to 30% noise. As the loading rate increased, the total system could not maintain the error extremely low as the noise increased – after this point, the error increased linearly. The storage capacities were ∼4% larger by setting ρ, i.e., sparsity term, to be smaller than 0.5.

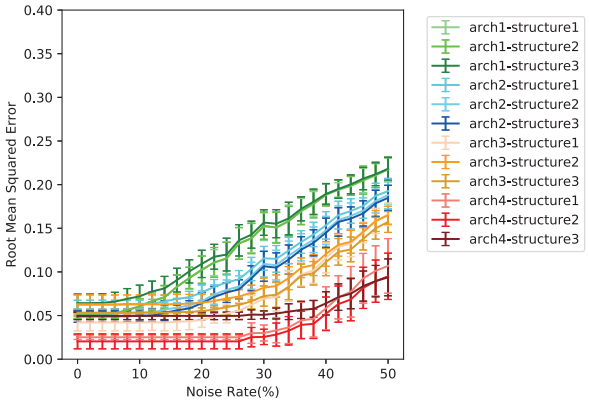

Fig. 13. Error on the output from the system of each architecture. The error bars indicate standard deviation.

3.3. Comparison with a Simple Autoencoder (Group C)

At 20% noise level, the errors of the proposed architecture were more than six times smaller than the errors of the autoencoder (**Fig. 14**). The trend of increasing the errors for the proposed architecture was the same as that in Group A, and the architecture output quite small errors up to 30%. The autoencoder, which processed three modalities, output larger errors when inputs contained errors. The trend of increasing the errors for the autoencoder was

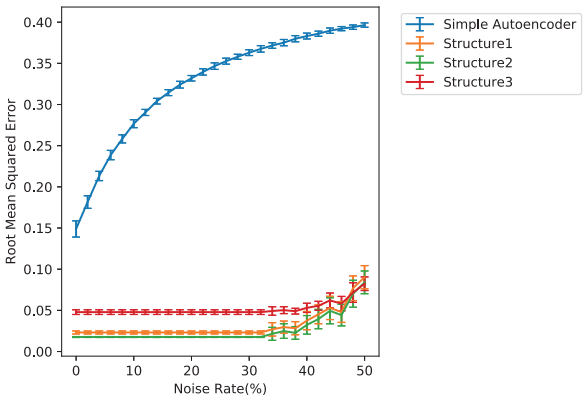

Fig. 14. Error on the output from the proposed architecture and an autoencoder when the input contains noise. The error bars indicate standard deviation.

similar to that of the logarithmic function.

It is clear that the trends of increasing the errors for the proposed architecture and the autoencoder differ significantly.

4. Discussion

Given these results, the validity of MACMSA as a robot system is discussed here in terms of cost, robustness, and best-effort optimized systems. Five conclusions can be drawn from the experimental results of Groups A, B, and C: 1) the proposed MACMSA architecture functions correctly according to the system design concept; 2) the encoder and decoder can be optimized using data with various averages and standard deviations; 3) the associator can store and recall accurately the patterns encoded by the encoder. 4) the system comprising encoders, decoders, and an associator can decrease errors better than a simple autoencoder does; and 5) the attributes are maintained even if the module sizes in the system are different.

Cost. If a simple autoencoder processes multimodal information, the number of neurons in the input layer is the dimensions of the modalities multiplied by the number of modalities. For instance, in the experiment for comparison with a simple autoencoder (Group C), the number of optimized weights between the input layer and Hidden layer 2 was $1{,}600 \times 1{,}200 = 1{,}920{,}000$, and the total number of optimized weights for the autoencoder was 5,760,000. Conversely, for the proposed architecture, the total number of optimized weights for Structures 1, 2, and 3 was 2,250,000, 562,500, and 22,500, respectively. Thus, the number of optimized weights for the proposed architecture is 2,835,000. The number of neurons comprising the simple autoencoder and the proposed architecture is the same. However, the total number of weights for the simple autoencoder is twice that for the proposed architecture. Therefore, the computational cost for the proposed architecture is lower than that for the simple autoencoder. Further, the computational cost to store the pattern to the Hopfield network is much lower than that to optimize the encoders and the decoder. Thus, it can be ignored. In addition, the computational cost to optimize the network increases exponentially because of the added modalities. For example, to add a new modality with 100 dimensions to the autoencoder in the experiment of Group C, 247,500 connections are added between the input layer and Hidden layer 2. Nevertheless, to add a new modality with 100 dimensions to the proposed architecture, only 7,500 connections are added between the input layer and Hidden layer 2, because only one additional independent autoencoder is optimized as an encoder and decoder. Thus, the computational cost to add a new modality on MACMSA is only a constant multiple.

In this study, offline learning and batch learning were used to reduce computational costs. Typically, online learning is considered necessary for optimizing robot systems, because once a robot becomes operational, it must remain operational. However, offline learning should be adequately practical in real-time by giving the robots time to optimize the system. Similarly, humans also take time to sort out information in the brain; this time is obtained during actions such as sleeping.

Srivastave and Salakhutdinov [29] used a deep Boltzmann machine as a generative model. Boltzmann machines potentially have a denoising function. The difference between a Boltzmann machine and a Hopfield network is that a Boltzmann machine has a stochastic behavior, whereas a Hopfield network has a deterministic behavior. A Hopfield network is advantageous because its cost for implementation is lower than that of a Boltzmann machine.

Robustness. In a real-world environment, noise is often added to the data because of changes in the environment or the state of the sensor. To consider such situations, in our experiments, some of the data were replaced by up to 50% noise. The results of Experiment 2-3 show that as the loading rate decreased, the range of noise levels over which extremely low errors can be maintained increased. Moreover, beyond that range, the errors increased linearly. It is assumed that the ability to reduce noise comes from the associator, because the encoders do not possess a strong ability to reduce noise. According to the results of Experiments 1-4 and 2-3, the decoders can output values close to those of the training data when the input is similar to the stored pattern. Thus, if the encoder has a strong ability to reduce noise, the number of errors in Experiments 1-2 and 2-1 should be low. However, in these experiments, which used only autoencoders, the number of errors was not negligible, and it increased non-linearly with respect to the noise levels in the input. In addition, at 50% noise, if the network size is small, the errors of the autoencoders and the proposed architecture are comparable. In contrast, if the network size is large, the errors in the proposed architecture are much fewer than those in the autoencoders. In real-world environments, the number of dimensions of the input from sensors is larger than that considered in this study. Thus, it

is easier to maintain errors low. For example, if the inputs are 64×64-pixel images, the network size of the encoder and decoder for these inputs is four times larger than the network size of the largest encoder and decoder used in this study. Focusing on the Hopfield network as an associator, the robustness against noise seems to be lower than the theoretical robustness. It is assumed that the stored patterns are not ideal patterns because they are generated by encoders. Therefore, it is important that the pattern formatted by an encoder be linearly independent. By improving the linear independence of the pattern, the associator can recall the correct pattern with a higher accuracy. Then, the error at the decoder can also be decreased against higher noise rates. In addition, in situations where the input includes a high level of noise (e.g., 50%), the input should be treated as another input derived from a different situation, because it seems unlikely that half of the input would be replaced by noise, even though it is true that noise is common in real-world environments. Thus, inputs including high levels of noise should be added to training data as new data. The scale of the Hopfield network used in Experiments 1-1–1-4 had the capacity to store approximately 100 patterns without sparse coding. Moreover, by precisely controlling sparseness, the storage capacity increases substantially; thus, it is easy to add new training data. Note that ρ was set as the target value of the firing rate for all the autoencoders, but the sparseness was not otherwise controlled in this study.

Best-effort optimized system. MACMSA is a simple architecture. It is composed of three blocks of encoder, associator, and decoder units. The encoders and decoders are optimized simultaneously as a single autoencoder for each mode. Moreover, when the autoencoders have been optimized, the pattern stored by the associator is determined uniquely. In other words, the entire system should be optimized when the autoencoders have been optimized. We could not find any published study on the importance of best-effort optimized systems for robotics control systems. However, things composed of parts that are individually the best are not always the best as a whole. Similarly, it is also not always true that a system composed of individually best modules is the best system. We believe that it is most important to optimize the combination of structures and modules. It is not necessary for structures and modules to be state-of-the-art. In fact, the proposed architecture is not composed of state-of-the-art components. However, the experiments indicate that the proposed architecture in practical use is robust against noise. A best-effort optimized system is also important for processing speed in real world. When a robot works in an actual environment, the system has to output control signals in real time. Even if the modules could process inputs rapidly, if a system composed of a combination of these modules takes time to output a control signal for the robot, it is useless. By considering a best-effort optimized system, it becomes possible to build a robot that works in a real environment and in real time. If it is necessary to add developmental functions in the next step, the following developments of two functions and one practical experiment would be appropriate. The first function is one that outputs a signal for time $t+1$ at time t. To control a robot, a system obtains the state at time t as inputs and outputs a signal to move a part of the robot's body to a position at time $t+1$. Storing the state of time $t+1$ as a pattern and recalling it using an associator is a potential way to output a signal that specifies the state at time $t+1$. The second potential function is precise control of the sparseness. Maximization of the storage capacity is important to cover a wide variety of possible states and allowable actions in a real environment. Therefore, controlling the sparseness is desirable, at least in an internal hidden layer that is the output layer of an encoder instead of the whole autoencoder.

Finally, the proposed method was implemented by using two different approaches. One consisted of writing original programs in C. In the programs, Eq. (2) is simply iterated a fixed number of times to optimize the networks. The other used PyTorch version 1.4.0 to calculate the gradients from Eq. (1) using automatic differentiation. Stable results were obtained using C programs but not with PyTorch. Therefore, all the results in this paper were obtained with programs implemented in C.

5. Conclusions

MACMSA was proposed to consider cost, robustness, and a best-effort optimized system. It is intended for robot control systems that process multimodal information. The robustness of MACMSA was evaluated by adding noise to the input. The results showed that MACMSA, which is designed according to the best-effort optimized system, is a suitable architecture, even if the input includes high levels of noise. The cost was discussed with a specific example, and best-effort optimized systems were also discussed. The design cost of the proposed architecture is lower than that of state-of-the-art methods. The calculation cost of the proposed architecture is also lower than that of a simple autoencoder that processes multimodal information concurrently.

Our future work will focus on implementing the proposed architecture and investigating its performance in real robots.

Implementing MACMSA with the functions mentioned in the discussion will take us one step closer to building robots that can truly interact with humans.

Acknowledgements

We appreciate the feedback offered by past and present members of our laboratory.

References:

[1] J. Mi and Y. Takahashi, "Humanoid robot motion modeling based on time-series data using kernel PCA and Gaussian process dynamical models," J. Adv. Comput. Intell. Intell. Inform., Vol.22, No.6, pp. 965-977, 2018.

[2] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent ASIMO: System overview and

integration," Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Vol.3, pp. 2478-2483, 2002.
[3] F. Tanaka, K. Isshiki, F. Takahashi, M. Uekusa, R. Sei, and K. Hayashi, "Pepper learns together with children: Development of an educational application," Proc. of the 2015 IEEE-RAS 15th Int. Conf. on Humanoid Robots (Humanoids), pp. 270-275, 2015.
[4] G. A. Bekey, "Autonomous robots: From biological inspiration to implementation and control," MIT Press, 2005.
[5] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, and P. Corke, "The limits and potentials of deep learning for robotics," Int. J. Rob. Res., Vol.37, Nos.4-5, pp. 405-420, 2018.
[6] D. Ramachandram and G. W. Taylor, "Deep multimodal learning: A survey on recent advances and trends," IEEE Signal Process. Mag., Vol.34, No.6, pp. 96-108, 2017.
[7] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Ng, "Multimodal deep learning," Proc. of the 28th Int. Conf. on Machine Learning (ICML'11), pp. 689-696, 2011.
[8] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," Proc. of 2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 1800-1807, 2017.
[9] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," Proc. of 2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 77-85, 2017.
[10] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," Int. J. Rob. Res., Vol.37, Issue 4-5, pp. 421-436, 2017.
[11] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," IEEE Access, Vol.6, pp. 14410-14430, 2018.
[12] Z. Chen, A. Jacobson, N. Sünderhauf, B. Upcroft, L. Liu, C. Shen, I. Reid, and M. Milford, "Deep learning features at scale for visual place recognition," Proc. of 2017 IEEE Int. Conf. on Robotics Automation (ICRA), pp. 3223-3230, 2017.
[13] S. Otsubo, Y. Takahashi, and M. Haruna, "Modular neural network for learning visual features, routes, and operation through human driving data toward automatic driving system," J. Adv. Comput. Intell. Intell. Inform., Vol.24, No.3, pp. 368-376, 2020.
[14] D. Kanda, S. Kawai, and H. Nobuhara, "Visualization method corresponding to regression problems and its application to deep learning-based gaze estimation model," J. Adv. Comput. Intell. Intell. Inform., Vol.24, No.5, pp. 676-684, 2020.
[15] H. A. Pierson and M. S. Gashler, "Deep learning in robotics: A review of recent research," Adv. Robot., Vol.31, Issue 16, pp. 821-835, 2017.
[16] K. Suzuki, H. Mori, and T. Ogata, "Motion switching with sensory and instruction signals by designing dynamical systems using deep neural network," IEEE Robot. Autom. Lett., Vol.3, No.4, pp. 3481-3488, 2018.
[17] N. Saito, K. Kim, S. Murata, T. Ogata, and S. Sugano, "Tool-use model considering tool selection by a robot using deep learning," Proc. of the 2018 IEEE-RAS 18th Int. Conf. on Humanoid Robots (Humanoids), pp. 270-276, 2018.
[18] J. Sergeant, N. Sünderhauf, M. Milford, and B. Upcroft, "Multimodal deep autoencoders for control of a mobile robot," Proc. of the Australasian Conf. on Robotics and Automation 2015 (ACRA 2015), pp. 1-10, 2015.
[19] K. Noda, H. Arie, Y. Suga, and T. Ogata, "Multimodal integration learning of robot behavior using deep neural networks," Rob. Auton. Syst., Vol.62, Issue 6, pp. 721-736, 2014.
[20] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," Int. J. Rob. Res., Vol.34, Issue 4-5, pp. 705-724, 2015.
[21] P.-C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata, "Repeatable folding task by humanoid robot worker using deep learning," IEEE Robot. Autom. Lett., Vol.2, No.2, pp. 397-403, 2017.
[22] M. Okada, "Notions of associative memory and sparse coding," Neural Netw., Vol.9, Issue 8, pp. 1429-1458, 1996.
[23] A. Billard, K. Dautenhahn, and G. Hayes, "Experiments on human-robot communication with Robota, an imitative learning communicating doll robot," B. Edmonds and K. Dautenhahn (Eds.), "Socially situated intelligence: A workshop held at SAB'98," pp. 4-16, University of Zürich Technical Report, 1998.
[24] S. Jockel, M. Mendes, J. Zhang, A. P. Coimbra, and M. Crisostomo, "Robot navigation and manipulation based on a predictive associative memory," Proc. of the 2009 IEEE 8th Int. Conf. on Development and Learning, 7pp., 2009.
[25] A. Ng, "Sparse Autoencoder," CS294A Lecture notes, https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf [accessed May 5, 2020]
[26] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," Proc. of the 2016 IEEE Int. Conf. on Robotics and Automation (ICRA), pp. 512-519, 2016.
[27] H. Asoh, "Deep representation learning by multi-layer neural networks," J. Jpn. Soc. Artif. Intell., Vol.28, No.4, pp. 649-659, 2013 (in Japanese).
[28] T. Okatani, "Deep learning," Kodansha, 2015 (in Japanese).
[29] N. Srivastava and R. Salakhutdinov, "Multimodal learning with deep Boltzmann machines," Proc. of the 25th Int. Conf. on Neural Information Processing Systems (NIPS'12), Vol.2, pp. 2222-2230, 2012.

Name:
Motohiro Akikawa

Affiliation:
Department of Computer Science, School of Computing, Tokyo Institute of Technology

Address:
Room 1710, J2 Building. 4259 Nagatsuta-cho, Midori-ku, Yokohama, Kanagawa 226-8503, Japan

Brief Biographical History:
2014 Master Degree, Tokyo Institute of Technology
2020 Withdrew from the doctorial course of Tokyo Institute of Technology
2020- Researcher, Tokyo Institute of Technology

Main Works:
• Artificial Intelligence, Neural Network

Name:
Masayuki Yamamura

Affiliation:
Department of Computer Science, School of Computing, Tokyo Institute of Technology

Address:
Room 1710, J2 Building. 4259 Nagatsuta-cho, Midori-ku, Yokohama, Kanagawa 226-8503, Japan

Brief Biographical History:
1989- Tokyo Institute of Technology
2000- Visiting Associate Professor, Binghamton University, State University of New York (SUNY)
2003- Guest Researcher, Institute of Physical and Chemical Research (RIKEN)
2009- Guest Professor, Tokyo Medical and Dental University

Main Works:
• R. Sekine et al, "Tunable synthetic phenotypic diversification on Waddington's landscape through autonomous signaling," Proc. Nati. Acad. Sci. U.S.A., Vol.108, No.44, pp. 17969-17973, 2011.

Membership in Academic Societies:
• Information Processing Society of Japan (IPSJ)
• The Japanese Society for Artificial Intelligence (JSAI)
• The Society of Instrument and Control Engineers (SICE)