2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Put-In-Box Task Generated from Multiple Discrete Tasks by a Humanoid Robot Using Deep Learning

Kei Kase[1], Kanata Suzuki[2], Pin-Chu Yang[3], Hiroki Mori[4], and Tetsuya Ogata[5]

Abstract— **For robots to have a wide range of applications, they must be able to execute numerous tasks. However, recent studies into robot manipulation using deep neural networks (DNN) have primarily focused on single tasks. Therefore, we investigate a robot manipulation model that uses DNNs and can execute long sequential dynamic tasks by performing multiple short sequential tasks at appropriate times. To generate compound tasks, we propose a model comprising two DNNs: a convolutional autoencoder that extracts image features and a multiple timescale recurrent neural network (MTRNN) to generate motion. The internal state of the MTRNN is constrained to have similar values at the initial and final motion steps; thus, motions can be differentiated based on the initial image input. As an example compound task, we demonstrate that the robot can generate a "Put-In-Box" task that is divided into three subtasks: open the box, grasp the object and put it into the box, and close the box. The subtasks were trained as discrete tasks, and the connections between each subtask were not trained. With the proposed model, the robot could perform the Put-In-Box task by switching among subtasks and could skip or repeat subtasks depending on the situation.**

I. Introduction

Robots that can combine appropriate actions in a dynamic environment to complete a complex task that comprises multiple subtasks are more flexible compared with those that cannot combine actions. Nowadays, most factories are not fully automated. Typically, in a changing work environment, e.g., small quantity production lines for customizable products, people perform various complicated tasks because robot manipulation using a modeling approach often requires experts to manually extract environmental features for various conditions and plan corresponding motion trajectories. As the number of situations and task types increase, it becomes extremely difficult to define the features and design appropriate motions for each condition. Furthermore, hard-coded robot manipulations are often unstable in uncontrolled environments. To address these problems, robot manipulation methods that can extract features easily, function in unknown situations, and generate and combine various tasks to suit changing work environments are desirable.

Deep neural networks (DNN) have improved performance in various fields [1] [2], and their characteristics are appealing for robot manipulation. DNNs can autonomously reduce high-dimensional data to low-dimensional data for feature extraction [3], and they can be used to classify unknown data from learned data [4]. These characteristics could be applied to extend robot manipulation by autonomously extracting environmental features and increasing generalizability. Previous studies have investigated robot manipulation with DNNs; however, few studies have addressed switching among multiple tasks via DNN-based robot manipulation.

Various types of DNNs, e.g., reinforcement learning (RL) and predictive learning, are used for object manipulation tasks. RL optimizes a motion trajectory by selecting the best motion from numerous attempts [5] [6]. In addition, RL may yield unforeseen optimal results for a robot or a task relative to optimizing motion trajectories. However, RL often requires sophisticated reward functions and robot interaction time for optimization. Predictive learning predicts the next sequence from previous steps. For example, recurrent neural networks (RNN), which are DNNs that use an internal memory to calculate the subsequent output from prior inputs, are used for predictive learning. RNNs can process sequential information and maintain robustness against instantaneous noise. Therefore, RNNs are suitable for robot manipulation and have been used for various tasks, such as drawing [7] and using tools [8].

We propose a model that uses two DNNs to generate multiple short discrete subtasks to complete a longer target task. To the best of our knowledge, few studies have investigated robot manipulation with DNNs to generate multiple shorter subtasks and switch among subtasks appropriately to generate a longer target task. In the proposed model, one DNN reduces the dimensionality of the image data autonomously, and the other, modified RNN model, generates motions by concatenating the reduced image data and the joint angles of the robot.

The proposed method utilizes a predictive learning model to learn policies based on a robotic demonstration with a human operator using a 3D mouse controller. With our model, the robot switches among short subtasks autonomously by recognizing images. Then, the robot executes the target task. By dividing a long target task into components, collecting and learning training data becomes more efficient. Further-

*This work was based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization and was supported by a Grant-in-Aid for Scientific Research (A) (No. 15H01710) from the Ministry of Education, Culture, Sports, Science, and Technology, Japan.

[1]Kei Kase, [2]Kanata Suzuki, [4]Hiroki Mori and [5]Tetsuya Ogata are with the Department of Intermedia Art and Science, Waseda University, Tokyo, Japan. kase@idr.ias.sci.waseda.ac.jp; suzuki@idr.ias.sci.waseda.ac.jp; mori@idr.ias.sci.waseda.ac.jp; ogata@waseda.jp

[3]Pin-Chu Yang is with the Department of Modern Mechanical and Engineering, Waseda University, Tokyo, Japan. komayang@sugano.mech.waseda.ac.jp

[1]Kei Kase, [2]Kanata Suzuki, [3]Pin-Chu Yang, and [5]Tetsuya Ogata are also with the Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology, Tokyo, Japan.

more, the proposed model enables the robot to repeat and skip individual subtasks appropriately.

II. Related Work

Many DNN-based robot manipulation studies use RL or predictive learning. Various tasks have been investigated relative to robot manipulation using RL [5] [6] [9] [10], [11], [12], [13], [14]. A previous study [14] focused on training multiple controllers and performing tasks in series; however, that study did not exploit visual inputs. Another study [10] focused on end-to-end visual servoing using RL. In that study, the robot demonstrated several types of task generation; however, the tasks were trained separately, i.e., multiple tasks were not trained simultaneously, and a switching phase was not considered.

For robot manipulation using predictive learning, several studies have trained multiple tasks simultaneously [15] [16]. For example, in one study [15], a robot was trained to interact with a box in various ways; however, a processed image was used to map the location of the box. The use of processed images limits the range of possible tasks, and image processing becomes more complicated as task complexity increases. In addition, numerous tasks have been generated using a multimodal DNN that integrates image features with the joint angles of a robot [16]. However, even though these frameworks deal with multiple tasks, they use consumer-grade robots to perform simple tasks, such as moving a box or rolling a ball, and do not consider switching tasks or complex tasks. One study employed a low-cost robotic arm to generate multiple tasks using LSTM [17]; however, the transitions between tasks were not considered. In another study, an industrial humanoid robot was trained to perform a more complicated task, i.e., folding a towel, [18] based on a previously proposed framework of [16]. However, generation of multiple tasks was not considered in that study.

The proposed model uses a framework that is similar to a previously proposed framework [16] for learning through sensory-motor integration, and we adopt an existing motion generation framework [15]. Furthermore, to switch among multiple tasks successively based on image data, we impose constraints to maintain a similar internal state of the RNN during the initial and final states of multiple tasks. The proposed model can learn multiple tasks simultaneously; thus, collecting training data becomes more efficient.

III. Methods

In this section, we describe the proposed model for generating and switching among short tasks based on image data. The proposed model uses two types of DNNs, i.e., a convolutional autoencoder (CAE) and a modified multiple timescale RNN (MTRNN), to extract image features and generate the robot's next motion based on previous images and motions (Fig. 1). The core of the proposed model is the modified MTRNN which ensures its internal state and the robot return to the similar state at the beginning and end of each subtask. This makes the transitions between multiple subtasks easier to execute.

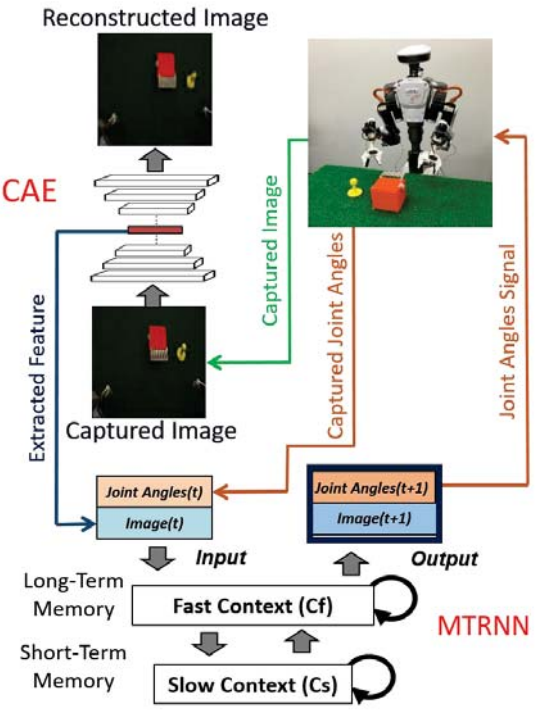


Fig. 1: Overview of the proposed motion generation model. An image is captured by the robot, and the CAE extracts the image feature. The captured joint angles are concatenated with the extracted image feature and used by the MTRNN to predict the next step. The predicted joint angles are then signaled to the robot to initiate movement.

A. Convolutional Autoencoder

In the proposed model, the CAE has features of both an AE and a convolutional NN (CNN) [19]. An AE can operate as an identity function that generates an output identical to the input using a bottleneck structure, which reduces the dimensionality of the data and reconstructs the original data from the reduced data. The most dimensionally reduced data at the middle layer are used as the extracted feature of the original data. Similarly, the proposed CAE is also an identity function; however, it reduces and reconstructs data using both convolutional and deconvolutional layers. Fully connected feedforward layers connect the convolutional and deconvolutional layers' kernels. CNNs used for image classification often contain pooling layers between each convolutional layer to perform nonlinear downsampling and obtain robustness against shifts and distortions. However, since positional information is essential for task generation, the pooling layer is removed in our CAE. The CAE is trained using mean squared error with the optimizer for the Adam (Adaptive Moment Estimation) algorithm [20].

B. Multiple Timescale RNN

As mentioned previously, RNNs have internal memory and can predict the next step from previous steps. This ability comes from the context layers, which have cyclic connections and compute the next output from the previous

context and input. Note that an MTRNN has multiple context layers with different time constants [15]. The proposed MTRNN has two context layers, i.e., a fast context (Cf) layer with a small time constant, and a slow context (Cs) layer with a large time constant. The Cf layer obtains more information from the current context, and the Cs layer obtains more information from the previous context. The MTRNN is similar to the LSTM since the Cs and Cf of the MTRNN function as the long and short term memory of the LSTM. We utilize the MTRNN for the sake of simpler and more direct analysis of internal representation

The forward calculation of the MTRNN is described in Eq. (1). The internal state of the ith neuron at time step t, i.e., $(u_i(t))$, is updated as follows:

$$u_i(t) = \begin{cases} \sum_{j \in I_{CF}} w_{ij} c_j(t) + b_i & (i \in I_O) \\ (1 - \frac{1}{\tau_i}) u_i(t-1) + \frac{1}{\tau_i} (\sum_{j \in I_I} w_{ij} x_j(t) & \\ \quad + \sum_{j \in I_{CF} \cup I_{CS}} w_{ij} c_j(t-1) + b_i) & (i \in I_{CF}) \\ (1 - \frac{1}{\tau_i}) u_i(t-1) & \\ \quad + \frac{1}{\tau_i} (\sum_{j \in I_{CF} \cup I_{CS}} w_{ij} c_j(t-1) + b_i) & (i \in I_{CS}), \end{cases} \tag{1}$$

where I_F, I_I, I_O, and I_S are the neuron index sets for the Cf, input, output, and Cs layers, respectively, τ_i is the time constant of the ith neuron, w_{ij} is the weight of the connection between the jth and ith neurons, $c_j(t)$ is the activation value of the jth context neuron at time step t, $x_j(t)$ is the jth external input at time step t, and b_i is the bias of the ith neuron. The respective activation values of context unit $c_i(t)$ and output unit $y_i(t)$ are calculated using $tanh$.

The MTRNN is trained to predict the data at time step $t+1$ from the input data at t using the mean squared error and the Adam optimizer. To switch among subtasks, the initial and final values of the contexts have similar values for all motions. This constraint is achieved by calculating the loss function for each task sequence as follows:

$$loss = \sum_{t=0}^{T} \|(\hat{y}(t) - y(t))\|^2 + \gamma (C(T) - C(0))^2, \tag{2}$$

where T is the total number of steps in task, $\hat{y}(t)$ and $y(t)$ are the training signal and predicted output, $C_p(t)$ and $C_q(t)$ are the values of Cf and Cs, respectively, and γ (set to 10) is a parameter that controls the loss of contexts. The constraint allows the network to form a point that we refer to as the strong attracting point, which is the point at which the internal states are likely to converge to and diverge from.

C. Training Phase and Motion Generation Method

For training, we first collect training data from a human experimenter manipulating the robot using a 3D mouse. As the robot is manipulated, the joint angles and image data are sampled. Then, the CAE is trained using the collected image data prior to training the MTRNN. To train the MTRNN, for time step t, the image features at t are extracted from the trained CAE and are concatenated with the joint angles at t. The concatenated data at t become the input of the MTRNN, which generates the output of t. The MTRNN's weights are optimized using loss function in Eq.(2). The output of the joint angle data of t becomes the input of $(t+1)$; however, the output of the image data of (t) is discarded. Here sampled data from the training data are used as the input of $(t+1)$.

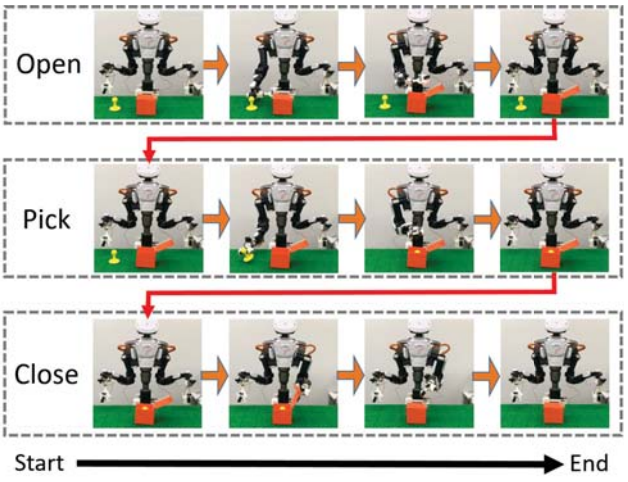
In motion generation (Fig.1), the robot obtains image data from its mounted camera, and the CAE is used to extract image feature. The extracted image feature is concatenated with the current joint angles and used as the input to the MTRNN. The predicted joint angles of the MTRNN are then used to signal the robot to move. After the robot moves, the image and joint angles are captured again and the next MTRNN input is prepared.

IV. Experiment

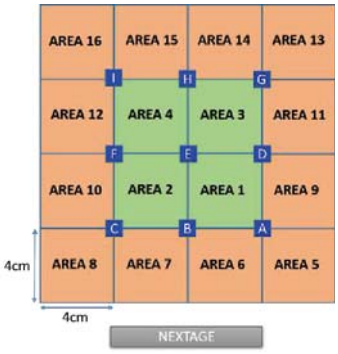
We designed an experiment to evaluate whether the proposed model can acquire transitions between the subtasks and demonstrate the beneficial aspects of generating longer sequential task from shorter sequential tasks. We used the Nextage Open Robot from Kawada Robotics [21] to investigate the effectiveness of the proposed robot manipulation model. The Nextage is a humanoid robot with a head-mounted camera and two arms, each of which has six degrees of freedom (DoF) and an attached gripper. We trained the Nextage on three subtasks (Fig. 2a): (1) opening a box (hereafter, "open"), (2) picking up an object and putting it into the box (hereafter, "pick"), and (3) closing the box (hereafter, "close"). By combining the three subtasks, the robot executed a complete "Put-In-Box" task. Each subtask was designed such that the initial and final positions of the robot were identical, thereby smoothing the transitions between each subtask. In addition, since the initial and final positions of the robot and the initial and final internal states of the MTRNN are identical, the only factor that causes transitions between subtasks is the differences in the image data. Each subtask was trained discretely, with no relation between the three subtasks.

The robot was trained to identify the box when it is placed at a fixed position and a stamp object at nine different positions. Each stamp position was 4 cm apart, forming a square (Fig. 2b). The training data were taken at these different positions three times for each subtask. In addition, for the pick subtask, the robot was trained with two additional objects: a white pepper container and white juice pack. A white steel can, salt container, yellow juice pack, and red juice pack were also prepared to test generalizability relative to picking up objects. The objects used in this experiment are shown in Fig. 2c.

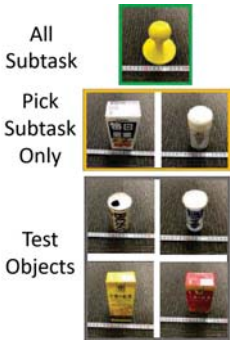
The robot's training data were created directly by the experimenter using the 3D mouse, and the motion and image data were sampled simultaneously. The training data for both the images and the joint angles were sampled at five frames per second, and each subtask required approximately 78 seconds to execute. A 64x64-pixel RGB image with 12,288 dimensions was captured by the robot's camera and



(a)



(b)



(c)

Fig. 2: (a) Third-person view of the robot generating motions for each subtask. The end of each subtask is the start of the next subtask, and the robot returns to the same position after completing a subtask.
(b) Positions where yellow stamp objects were learned. The blue squares are the nine learned positions.
(c) Objects learned and tested in the experiment. Blue border: objects trained for all subtasks; yellow border: objects trained for only the pick subtask; red border: objects not learned.

processed by the CAE to reduce its dimensionality to 20. The training data for the MTRNN were the data of the 20 original image features concatenated with 14 joint angles (i.e., the DoF) of the robot's arms and grippers. The training data for all subtasks were looped three times, meaning that the training data were prepared such that each subtask was repeated three times to stabilize and smooth the output.

The CAE was trained over 1,500,000 iterations with a learning rate of α = 0.0002, β_1 = 0.90, β_2 = 0.999, and $\varepsilon = 10^{-8}$. Here, the mini-batch size was 100. The MTRNN was trained over 150,000 epochs with a learning rate of α = 0.001, β_1 = 0.90, β_2 = 0.999, and $\varepsilon = 10^{-8}$. Here, the weight decay was 10^{-3}. Both the CAE and the MTRNN are optimized with the Adam. Other details about the CAE and MTRNN used in this study are shown in Table I.

TABLE I: Neural Network Parameters

DCNN	conv@3ch - conv@32ch - conv@64ch - conv@128ch - conv@256ch - full@1000 - full@20 - full@1000 - dconv@256ch - dconv@128ch - dconv@64ch - dconv@32ch - dconv@3ch
MTRNN	Input_nodes@34 - Cf_nodes@100 - Cs_nodes@5 Cf_τ@5 , Cs_τ@70

conv: convolutional layer dconv: deconvolutional layer
full: fully-connected layer

V. Results and Discussion

A. Extracting Image Features

The features extracted by the CAE must be distinguishable for each subtask for the robot to switch between subtasks. Figure 3 shows the results for the extracted features of the initial step of the three subtasks processed by principal component analysis (PCA) to reduce the dimensionality of the data linearly for visualization. The contribution ratios of the first and second PCs were 16.84% and 11.85%, respectively. The images from the open subtask are clearly separated from images of the pick and close subtasks, and the images from the pick and close subtasks are clustered. This occurs because, compared to the box, the stamp is relatively small in the captured image. These differences in the extracted features are essential for transitioning between the subtasks because the internal state of the MTRNN and the robot joint angles have similar if not identical values for each subtask; thus, only the image is different.

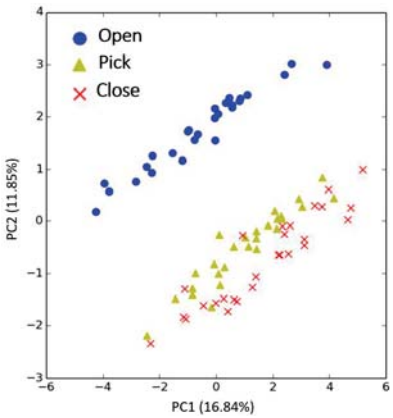


Fig. 3: PCA of features extracted by the CAE at the initial step of the three subtasks (blue: open; yellow: pick; red: close). The clusters for each subtask are identifiable, which indicates that the CAE autonomously extracted image features, thereby demonstrating its classification ability.

B. Generation of Put-In-Box Task

We tested with the robot to determine if the proposed model can generate multiple tasks that are appropriate for the given situation. The robot autonomously generated the entire Put-In-Box task when the yellow stamp and closed box were placed for the open subtask, even though each subtask was trained separately. Figure 4 shows the PCA results for the context layers of the MTRNN during motion generation.

The results show that the context layers start and end at similar values, thereby forming the attracting point. Each subtask starts at the strong attracting point created by the constraint on the MTRNN and diverges from this point using the difference of the MTRNN's input. Since the difference of the input at the start of each subtask is the image data, the data are used to diverge from the attracting point, generate a motion corresponding to the image data, and converge back to the attracting point.

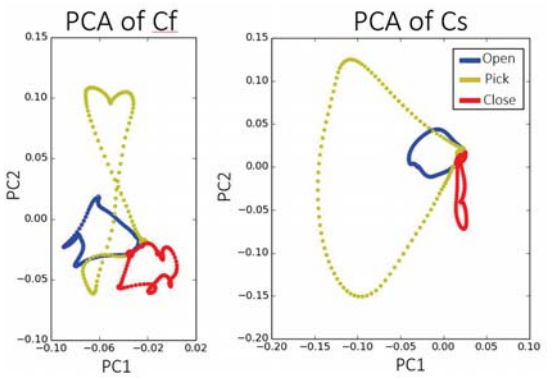


Fig. 4: PCA of context layers of the MTRNN during generation of the Put-In-Box task. Each subtask is represented by a different color, and each subtask begins and ends its motion at a similar value.

We also tested the MTRNN without constraints to evaluate the effects of constraints on the MTRNN. The unconstrained MTRNN was prepared by training using Eq.(2), where γ was set to zero but trained with the same initial value of the MTRNN. We evaluated the differences between the constrained and unconstrained MTRNNs by comparing the success rates of generating complete Put-In-Box tasks with the yellow stamp. In five trials, the robot with the unconstrained MTRNN failed to generate a complete task. This robot generated the open and pick subtasks successfully; however, it failed to switch to the close subtask. Here, the robot repeated the pick subtask instead. In contrast, the robot with the constrained MTRNN generated the complete task for all five trials.

The model of training multiple discrete motions with constraints allows the robot to execute motions that are difficult when trained consecutively. One benefit of this is the realization of a reiteration ability, which allows the robot to perform the same task repeatedly in shorter segments. A previous study [18] has demonstrated the reiteration ability of robot manipulation tasks after an entire sequence is completed. The proposed model allows reiteration after each short task and results in faster recovery from a failed task execution. In addition, the proposed method allows the robot to skip subtasks, thereby creating an opportunity for easier human interaction and interruption, e.g., when the experimenter performs pick subtask after the robot completes the open subtask, the robot autonomously starts the close subtask.

C. Position Generalizability

The robot was tested relative to executing the pick subtask at the areas shown in Fig. 2b. Each area was tested five times with the yellow stamp object. The robot could pick up the object flawlessly at areas 1 to 4 (i.e., areas within the training positions). The robot was also able to pick up the object at areas 5 to 16 (i.e., areas outside the training positions), but with failed attempts. The total success rate of picking up the object in 16 areas was 88.75% (80 attempts).

D. Generalization of Objects

Compared to motion generation models that train tasks as a whole, the proposed model allows subtasks to be trained separately, which reduces the data collection. The proposed model can control the numbers of datasets to be learned depending on the task generation difficulty. Therefore, it is possible to increase the number of datasets for tasks that require high precision control (and vice versa). This characteristic increases the efficiency of data collection when training less demanding tasks. In our experiment, the pick subtask required more precise control to grasp the object, and the open and close subtasks required less precision. Therefore, to evaluate the benefit of training subtasks separately, we prepared a model trained using only the yellow stamp object. The reconstructed images obtained using the CAE with only the yellow stamp and the CAE with the object added at the start of the pick subtask are shown in Fig. 5.

The CAE trained with only the yellow stamp could not reconstruct the images of the additional objects clearly, and this affected the precision of the robot's motion generation. To confirm the effects of the CAE, the robot's ability to generate pick subtasks was tested with all objects. For the red juice pack, the robot was unable to recognize when it was required to perform the pick subtask. Here, it generated the close subtask instead. For the other objects, the robot generated the pick subtask but failed to pick up the untrained objects (it knocked over the objects with its grippers). In contrast, the CAE trained with additional objects could reconstruct additional untrained objects more clearly and more vividly compared with the CAE trained with only the yellow stamp object. With the CAE trained with additional objects, the robot generated the pick subtask for all objects and could pick the objects up without knocking them over.

With the CAE trained with additional objects, the robot was tested to determine whether it could generate complete Put-In-Box tasks with objects trained using only the pick subtask and objects that were not learned for any task. The robot generated complete Put-In-Box tasks successfully for additional objects trained for just the pick subtask, even though the robot was not trained to generate the open or close subtasks with these objects. Furthermore, the robot could generate complete tasks with all untrained objects except for the red juice pack. For this object, the robot successfully generated the open and pick subtasks but failed to switch to the close subtask, instead repeating the pick subtask.

With the proposed model, the robot successfully increased robustness against objects for the pick subtask by only

increasing the number of training datasets for the pick subtask, which reduced the data collection and network training burden.

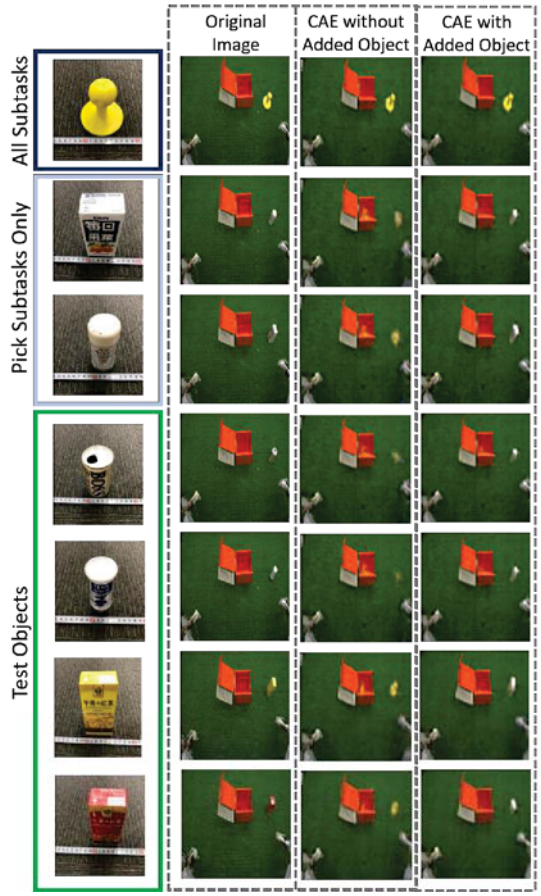


Fig. 5: Image reconstruction of the CAE trained with only the stamp object and the CAE trained with additional objects for the pick subtask. The CAE trained with additional objects could reconstruct the original image more precisely than the other CAE.

VI. Conclusion

In this paper, we have proposed using a sensory-motor integrated DNN model to realize a humanoid robot capable of continuously switching among short sequential subtasks based solely on image data in order to complete a longer sequential task. Since the number of related studies into robots performing multiple tasks with DNNs is limited, we selected the Put-In-Box task as an example to demonstrate the robot's ability to switch among subtasks and the benefit of training subtasks discretely. The proposed model executed the Put-In-Box task successfully using the advantageous characteristics of DNNs, i.e., feature extraction of high-dimensional data and generalization.

In future, we plan to investigate the benefits of training shorter sequences separately by adding more objects to be picked up. We also plan to train more complex and more extensive tasks that require reiteration ability, such as cutting food.

References

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12, 2015, pp. 1–9.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. Den, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the Game of Go with Deep Neural Networks and Tree Search," *Nature*, vol. 529, no. 1, pp. 1–37, 2016.

[3] G. Hinton and R. R. Salakhutdinov, "Science," *Science*, vol. 313, no. July, pp. 504–507, 2006.

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

[5] T. Lampe and M. Riedmiller, "Acquiring visual servoing reaching and grasping skills using neural reinforcement learning," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–8.

[6] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection," *The International Journal of Robotics Research*, p. 027836491771031, 2016.

[7] K. Sasaki, H. Tjandra, K. Noda, K. Takahashi, and T. Ogata, "Neural network based model for visual-motor integration learning of robot's drawing behavior: Association of a drawing motion from a drawn image," in *IEEE International Conference on Intelligent Robots and Systems*, 2015, pp. 2736–2741.

[8] K. Takahashi, T. Ogata, and H. Tjandra, "Tool body Assimilation Model Based on Body Babbling and Neuro-dynamical System," *International Conference on Artificial Neural Networks*, vol. 2015, pp. 115–127, 2014.

[9] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning Force Control Policies for Compliant Robotic Manipulation," *International Conference on Machine Learning*, pp. 4639–4644, 2012.

[10] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," *ArXiv:1504.00702*, 2015.

[11] S. Lange, M. Riedmiller, and A. Voigtländer, "Autonomous reinforcement learning on raw visual input data in a real world application," *Proceedings of the International Joint Conference on Neural Networks*, pp. 10–15, 2012.

[12] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images," *arXiv:1506.07365*, pp. 1–9, 2015.

[13] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Björkman, "Deep Predictive Policy Training using Reinforcement Learning," *arXiv:1703.00727*, 2017.

[14] W. Han, S. Levine, and P. Abbeel, "Learning compound multi-step controllers under unknown dynamics," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-December, pp. 6435–6442, 2015.

[15] Y. Yamashita and J. Tani, "Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment," *PLoS Computational Biology*, vol. 4, no. 11, 2008.

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

[17] R. Rahmatizadeh, P. Abolghasemi, L. Boloni, and S. Levine, "Vision-Based Multi-Task Manipulation for Inexpensive Robots Using End-To-End Learning from Demonstration," *arXiv:1707.02920*, 2017.

[18] 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 Robotics and Automation Letters*, vol. 2, no. 2, pp. 397–403, 2017.

[19] J. Masci, U. Meier, D. Ciresan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Lecture Notes in Computer Science*, vol. 6791 LNCS, no. PART 1, 2011, pp. 52–59.

[20] D. P. Kingma and J. L. Ba, "Adam: A Method for Stocastic Optimization," *arXiv:1412.6980v9*, pp. 1–15, 2015.

[21] K. Robotics, "Next Generation Industrial Robot Nextage." [Online]. Available: http://nextage.kawada.jp/en/