

Deep Learning Scooping Motion using Bilateral Teleoperations

Hitoe Ochi, Weiwei Wan, Yajue Yang, Natsuki Yamanobe, Jia Pan, and Kensuke Harada

Abstract— We present bilateral teleoperation system for task learning and robot motion generation. Our system includes a bilateral teleoperation platform and a deep learning software. The deep learning software refers to human demonstration using the bilateral teleoperation platform to collect visual images and robotic encoder values. It leverages the datasets of images and robotic encoder information to learn about the inter-modal correspondence between visual images and robot motion. In detail, the deep learning software uses a combination of Deep Convolutional Auto-Encoders (DCAE) over image regions, and Recurrent Neural Network with Long Short-Term Memory units (LSTM-RNN) over robot motor angles, to learn motion taught by human teleoperation. The learnt models are used to predict new motion trajectories for similar tasks. Experimental results show that our system has the adaptivity to generate motion for similar scooping tasks. Detailed analysis is performed based on failure cases of the experimental results. Some insights about the cans and cannots of the system are summarized.

I. INTRODUCTION

Common household tasks require robots to act intelligently and adaptively in various unstructured environment, which makes it difficult to model control policies with explicit objectives and reward functions. One popular solution [1] [2] is to circumvent the difficulties by learning from demonstration (LfD). LfD allows robots to learn skills from successful demonstrations performed by manual teaching. In order to take advantage of LfD, we develop a system which enables human operators to demonstrate with ease and enables robots to learn dexterous manipulation skills with multi-modal sensed data. Fig.1 shows the system. The hardware platform of the system is a bi-lateral tele-operation systems composed of two same robot manipulators. The software of the system is a deep neural network made of a Deep Convolutional Auto-Encoder (DCAE) and a Recurrent Neural Network with Long Short-Term Memory units (LSTM-RNN). The deep neural network leverages the datasets of images and robotic encoder information to learn about the inter-modal correspondence between visual images and robot motion, and the proposed system use the learnt model to generate new motion trajectories for similar tasks.

Using the system, we can conduct experiments of robot learning different basic behaviours using deep learning algorithms. Especially, we focus on a scooping task which is common in home kitchen. We demonstrate that our system

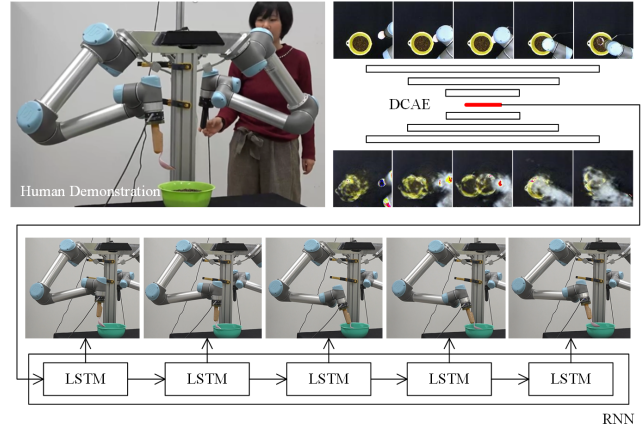


Fig. 1: The bilateral teleoperation system for task learning and robotic motion generation. The hardware platform of the system is a bi-lateral tele-operation systems composed of two same robot manipulators. The software of the system is a deep neural network made of a Deep Convolutional Auto-Encoder (DCAE) and a Recurrent Neural Network with Long Short-Term Memory units (LSTM-RNN). The deep learning models are trained by human demonstration, and used to generate new motion trajectories for similar tasks.

has the adaptivity to predict motion for a broad range of scooping tasks. Meanwhile, we examine the ability of deep learning algorithms with target objects placed at different places and prepared in different conditions. We carry out detailed analysis on the results and analyzed the reasons that limited the ability of the proposed deep learning system. We reach to a conclusion that although LfD using deep learning is applicable to a wide range of objects, it still requires a large amount data to adapt to large varieties. Mixed learning and planning is suggested to be a better approach.

The paper is organized as follows. Section 2 reviews related work. Section 3 presents the entire LfD system including the demonstration method and the learning algorithm. Section 4 explains how the robot perform a task after learning. Section 5 describes and analyzes experiment setups and results. Section 6 draws conclusions and discusses possible methods to improve system performance.

II. RELATED WORK

The learning method we used is DCAE and RNN. This section reviews their origins and state-of-the-art applications.

Auto-encoders were initially introduced by Rumelhart et al. [3] to address the problem of unsupervised back propaga-

Hitoe Ochi, Weiwei Wan, and Kensuke Harada are with Graduate School of Engineering Science, Osaka University, Japan. Natsuki Yamanobe, Weiwei Wan, and Kensuke Harada are also affiliated with Natioanl Institute of Advanced Industrial Science and Technology (AIST), Japan. Yajue Yang and Jia Pan are with City University of Hongkong, China. E-mail: wan@sys.es.osaka-u.ac.jp

tion. The input data was used as the teacher data to minimize reconstruction errors [4]. Auto-encoders were embedded into deep neural network as DCAE to explore deep features in [5] [6] [7] [8], etc. DCAE helps to learn multiple levels of representation of high dimensional data.

RNN is the feed-backward version of conventional feed forward neural network. It allows the output of one neuron at time t_i to be input of a neuron for time t_{i+1} . RNN may date back to the Hopfield network [9]. RNN is most suitable for learning and predicting sequential data. Some successful applications of RNN include handwriting recognition [10], speech recognition [11], visual tracking [12], etc. RNN has advantages over conventional mathematical models for sequential data like Hidden Markov Model (HMM) [13] in that it uses scalable historical characters and is applicable to sequences of varying time lengths.

A variation of RNN is Multiple Timescale RNN (MTRNN), which is the multiple timescale version of traditional RNN and was initially proposed by Yamashita and Tani [14] to learn motion primitives and predict new actions by combining the learnt primitives. The MTRNN is composed of multiple Continuous Recurrent Neural Network (CTRNN) layers that allow to have different timescale activation speeds and thus enables scalability over time. Arie et al. [15] Jeong et al. [16] are some other studies that used MTRNN to generate robot motion.

RNN-based methods suffers from a vanishing gradient problem [17]. To overcome this problem, Hochreiter and Schmidhuber [18] developed the Long Short Term Memory (LSTM) network. The advantages of LSTM is it has an input gate, an output gate, and a forget gate which allow the cells to store and access information over long periods of time.

The recurrent neural network used by us in this paper is RNN with LSTM units. It is proved that RNN with LSTM units are effective and scalable in long-range sequence learning [19]¹. By introducing into each LSTM unit a memory cell which can maintain its state over time, LSTM network is able to overcome the vanishing gradient problem. LSTM is especially suitable for applications involving long-term dependencies [21].

Together with the DCAE, we build a system allowing predicting robot trajectories for diverse tasks using vision systems. The system uses bilateral teleoperation to collect data from human beings, like a general LfD system. It trains a DCAE as well as a LSTM-RNN model, and use the model to learn robot motions to perform similar tasks. We performed experiments by especially focusing on a scooping task that is common in home kitchen. Several previous studies like [2] [22] [23] [24] also studied learning to perform similar robotic tasks using deep learning models. Compared with them, we not only demonstrate the generalization of deep models in robotic task learning, but also carry out detailed analysis on the results and analyzed the reasons that

limited the ability of the proposed deep learning system. Readers are encouraged to refer to the experiments and analysis section for details.

III. THE SYSTEM FOR LfD USING DEEP LEARNING

A. The bilateral teleoperation platform

Our LfD system utilizes bilateral teleoperation to allow human operators to adaptively control the robot based on his control. Conventionally, teleoperation was done in master-slave mode by using a joystick [23], a haptic device [25], or a virtual environment [24] as the master device. Unlike the conventional methods, we use a robot manipulator as the master. As Figure 2 shows, our system is composed of two identical robot systems comprising a Universal Robot 1 arm at the same joint configuration and a force-torque sensor attached to the arms end-effector. The human operator drags the master at its end-effector and the controller calculates 6 dimensional Cartesian velocity commands for robots to follow the human operators guidance. This dual-arm bilateral teleoperation system provides similar operation spaces for the master and slave, which makes it more convenient for the human operator to drag the master in a natural manner. In addition, we install a Microsoft Kinect 1 above the slave manipulator to capture depth and RGB images of the environment.

The bilateral teleoperation platform provides the human operator a virtual sense of the contact force to improve LfD [26]. While the human operator works on the master manipulator, the slave robot senses a contact force with a force-torque sensor installed at its wrist. A controller computes robot motions considering both the force exerted by human beings and the force feedback from the force sensor. Specifically, when the slave does not contact with the environment, both the master and slave move following human motion. When the slave has contact feedback, the master and slave react considering the impedance from force feedback. The human operator, meanwhile, would feel the impedance from the device he or she is working on (namely the master device) and react accordingly.

B. The deep learning software

LSTM-RNN supports both input and output sequences with variable length, which means that one network may be suitable for varied tasks with different length of time. Fig.2 illustrates a LSTM recurrent network which outputs prediction.

The data of our LfD system may include an image of the environment, force/torque data sensed by a F/T sensor installed at the slaves end-effector, robot joint positions, etc. These data has high dimensionality which makes computation infeasible. To avoid the curse of dimensionality, we use DCAE to represent the data with auto-selected features. DCAE encodes the input data with a encoder and reconstructs the data from the encoded values with a decoder. Both the encoder and decoder could be multi-layer

¹There are some work like [20] that used MTRNN with LSTM units to enable multiple timescale scalability.

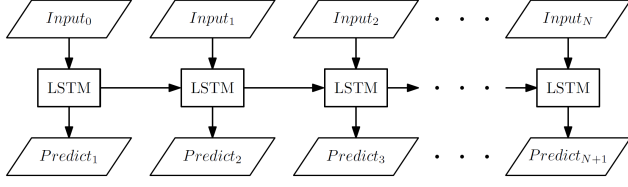


Fig. 2: LSTM-RNN: The subscripts in $Input_i$ and $Predict_i$ indicate the time of inputs and for which predictions are made. An LSTM unit receives both current input data and hidden states provided by previous LSTM units as inputs to predict the next step.

convolutional networks shown in Fig.3. DCAE is able to properly encode complex data through reconstruction, and extract data features and reduce data dimension.

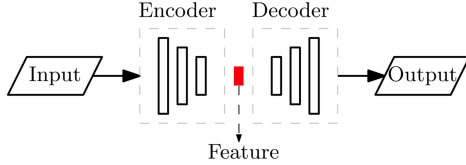


Fig. 3: DCAE encodes the input data with a encoder and reconstructs the data from the encoded values with a decoder. The output of DCAE (the intermediate layer) is the extracted data features.

The software of the LfD system is a deep learning architecture composed of DCAE and LSTM-RNN. The LSTM-RNN model is fed with image features computed by the encoder and other data such as joint positions, and predicts a mixture of the next motion and surrounding situation. The combination of DCAE and LSTM-RNN is shown in Fig.4.

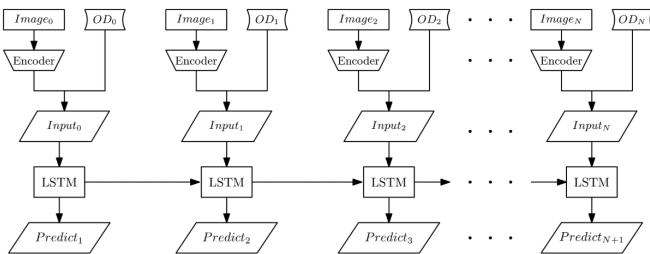


Fig. 4: The entire learning architecture. OD means Other Data. It could be joint positions, force/torque values, etc. Although drawn as a single box, the model may contain multiple LSTM layers.

IV. LEARNING AND PREDICTING MOTIONS

A. Data collection and training

The data used to train DCAE and LSTM-RNN is collected by bilateral teleoperation. The components of the bilateral teleoperation platform and the control diagram of LfD using the bilateral teleoperation platform are shown in Fig.5. A

human operator controls a master arm and performs a given task by considering force feedback ($F_h \oplus F_e$ in the figure) from the slave side. As the human operator moves the master arm, the Kinect camera installed at the middle of the two arms take a sequence of snapshots as the training images for DCAE. The motor encoders installed at each joint of the robot take a sequence of 6D joint angles as the training data of LSTM-RNN. The snapshots and changing joint angles are shown in Fig.6. Here, the left part shows three sequences of snapshots. Each sequence is taken with the bowl placed at a different position (denoted by $pos1$, $pos2$, and $pos3$ in the figure). The right part shows a sequence of changing joint angles taught by the human operator.

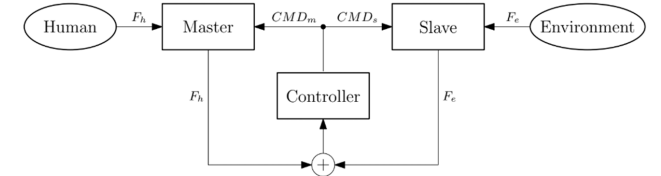
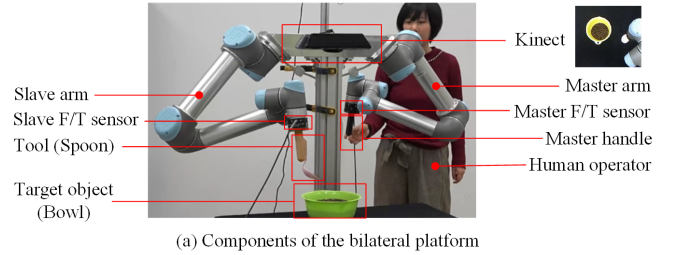


Fig. 5: Bilateral Teleoperation Diagram.

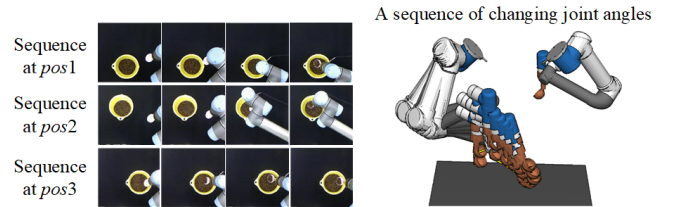


Fig. 6: The data used to train DCAE and LSTM-RNN. The left part shows the snapshot sequences taken by the Kinect camera. They are used to train DCAE. The right part shows the changing joint angles taught by human operators. They are used to further train LSTM-RNN.

B. Generating robot motion

After training the DCAE and the LSTM-RNN, the models are used online to generate robot motions for similar tasks. The trajectory generation involves a real-time loop of three phases: (1) sensor data collection, (2) motion prediction, and (3) execution. At each iteration, the current environment information and robot state are collected and processed and then attached to the sequence of previous data. The current robot state is fed to the pre-trained LSTM-RNN model to

predict next motions that a manipulator uses to take action. In order to ensure computational efficiency, we keep each input sequence in a queue with fixed length.

The process of training and prediction is shown in Fig.1. Using the pre-trained DCAE and LSTM-RNN, the system is able to generate motion sequences to perform similar tasks.

V. EXPERIMENTS AND ANALYSIS

We use the developed system to learn scooping tasks. The goal of this task is to scoop materials out from a bowl placed on a table in front of the robot (see Fig.1). Two different bowls filled with different amount of barley are used in experiments. The two bowls include a yellow bowl and a green bowl. The volumes of barley are set to “high” and “low” for variation. In total there are $2 \times 2 = 4$ combinations, namely {“yellow” bowl-“low” barley, “yellow” bowl-“high” barley, “green” bowl-“low” barley, and “green” bowl-“high” barley}. Fig.7(a) shows the barley, the bowls, and the different volume settings. During experiments, a human operator performs teleoperated scooping as he/she senses the collision between the spoon and the bowl after the spoon is inserted into the materials. Although used for control, the F/T data is not fed into the learning system, which means the control policy is learned only based on robot states and 2D images.

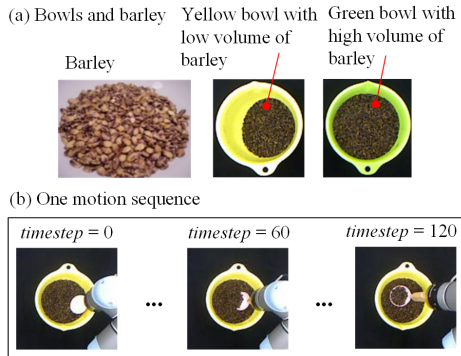


Fig. 7: (a) Two different bowls filled with different amount of barley are used in experiments. In total, there are $2 \times 2 = 4$ combinations. (b) One sequence of scooping motion.

The images used to train DCAE are cropped by a 130×130 window to lower computational cost. The DCAE has 2 convolutional layers with filter sizes of 32 and 16, followed by 2 fully-connected layers of sizes 100 and 10. The decoder has exactly the same structure. LeakyReLU activation function is used for all layers. Dropout is applied afterwards to prevent over fitting. The computation is performed on a Dell T5810 workstation with Nvidia GTX980 GPU.

A. Experiment 1: Same position with RGB/Depth images

In the first group of experiments, we place the bowl at the same position, and test different bowls with different amounts of contents. In all, we collect 20 sequences of data with 5 for each bowl-content combination. Fig.7(b) shows one sequence of the scooping motion. We use 19 of the 20 sequences of

data to train DCAE and LSTM-RNN and use the remaining one group to test the performance.

Parameters of DCAE is as follows: Optimization function: Adam; Dropout rate: 0.4; Batch size: 32, Epoch: 50. We use both RGB images and Depth images to train DCAE. The pre-trained models are named RGB-DCAE and Depth-DCAE respectively. Parameters of LSTM-RNN is: Optimization function: Adam; Batch size: 32; Iteration: 3000.

The results of DCAE is shown in Fig.8(a). The trained model is able to reconstruct the training data with high precision. Readers may compare the first and second rows of Fig.8(a.1) for details. Meanwhile, the trained model is able to reconstruct the test data with satisfying performance. Readers may compare the first and second row of Fig.8(a.2) to see the difference. Although there are noises on the second rows, they are acceptable.

The results of LSTM-RNN show that the robot is able to perform scooping for similar tasks given the RGB-DCAE. However, it cannot precisely differ “high” and “low” volumes. The results of LSTM-RNN using Depth-DCAE is unstable. We failed to spot a successful execution. The reason depth data is unstable is probably due to the low resolution of Kinect’s depth sensor. The vision system cannot differ if the spoon is at a pre-scooping state or post-scooping state, which makes the robot hard to predict next motions.

B. Experiment 2: Different positions

In the second group of experiments, we place the bowl at different positions to further examine the generalization ability of the trained models.

Similar to experiment 1, we use bowls with two different colors (“yellow” and “green”), and used two different volumes of contents (“high” and “low”). The bowls were placed at 7 different positions. At each position, we collect 20 sequences of data with 5 for each bowl-barley combination. In total, we collect 140 sequences of data. 139 of the 140 sequences of data are used to train DCAE and LSTM-RNN. The remaining 1 sequence are used for testing. The parameters of DCAE and LSTM-RNN are the same as experiment 1.

The results of DCAE are shown in Fig.8(b). The trained model is able to reconstruct the training data with high precision. Readers may compare the first and second rows of Fig.8(b.1) for details. In contrast, the reconstructed images show significant difference for the test data. It failed to reconstruct the test data. Readers may compare the first and second row of Fig.8(b.2) to see the difference. Especially for the first column of Fig.8(b.2), the bowl is wrongly considered to be at a totally different position.

The LSTM-RNN model is not able to generate scooping motion for either the training data or the test data. The motion is randomly changing from time to time. It doesn’t follow any pre-taught sequences. The reason is probably the bad reconstruction performance of DCAE. The system failed to correctly find the positions of the bowls using the encoded

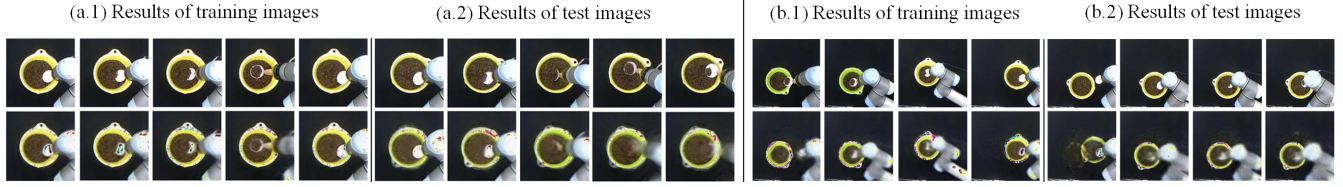


Fig. 8: The results of DCAE for experiment 1 and 2. (a.1-2) are the results of training data and test data for experiment 1. (b.1-2) are the results of training data and test data for experiment 2.

features. Based on the analysis, we increase the training data of DCAE in Experiment 3 to improve its reconstruction.

C. Experiment 3: Increasing the training data of DCAE

The third group of experiments has exactly the same scenario settings and parameter settings as Experiment 2, except that we use planning algorithms to generate scooping motion and collect more scooping images.

The new scooping images are collected following the work flow shows in Fig.9. We divide the work space into around 100 grids, place bowl at these places, and sample arm boatwains and orientations at each of the grid. In total, we additionally generate $100 \times 45 \times 3 = 13500$ (12726 exactly) extra training images to train DCAE. Here, “100” indicates the 100 grid positions. “45” and “3” indicate the 45 arm positions and 3 arm rotation angles sampled at each grid.

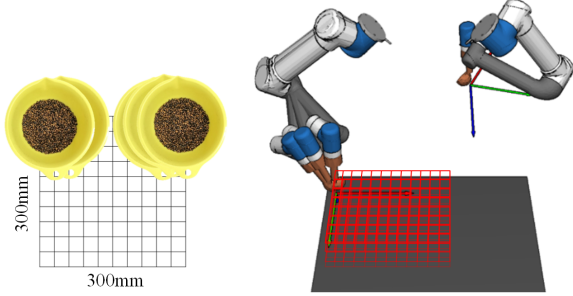


Fig. 9: Increase the training data of DCAE by automatically generating motions across a 10×10 grids. In all, $100 \times 45 \times 3 = 13500$ extra training images were generated. Here, “100” indicates the 100 grid positions. At each grid, 45 arm positions and 3 arm rotation angles are sampled.

The DCAE model is trained with the 140 sequences of data in experiment 2 (that is $140 \times 120 = 16800$ images, 17714 exactly), together with the 13500 extra images collected using the planner. The parameters of DCAE are exactly the same as Experiment 1 and 2. The results of DCAE is shown in Fig.10. Compared with experiment 2, DCAE is more stable. It is able to reconstruct both the training images and the test images with satisfying performance, although the reconstructed spoon position in the sixth and seventh columns of Fig.10(b) have relatively large offsets from the original image.

The trained DCAE model is used together with LSTM-RNN to predict motions. The LSTM-RNN model is trained

using different data to compare the performance. The results are shown in Table.I. Here, A1-A7r.s1 indicates the data used to train DCAE and LSTM-RNN. The left side of “_” shows the data used to train DCAE. A1-A7 means all the sequences collected at the seven bowl positions in experiments 2 are used. r means the additional data collected in experiment 3 is used. The right side of “_” shows the data used to train LSTM-RNN. s1 means only the sequences at bowl position s1 are used to train LSTM-RNN. s1s4 means both the sequences at bowl position s1 and position s4 are used to train LSTM-RNN.

The results show that the DCAE trained in experiment 3 is able to predict motion for bowls at the same positions. For example, (row position s1, column A1-A7r.s1) is \bigcirc , (row position s1, column A1-A7r.s1s4) is also \bigcirc . The result, however, is unstable. For example, (row position s2, column A1-A7r.s1s4) is \times , (row position s4, column A1-A7r.s4) is also \times . The last three columns of the table shows previous results: The A1_s1 column and A4_s4 column correspond to the results of experiment 1. The A1A4_s1s4 column correspond to the result of experiment 2.

Results of the three experiments show that the proposed model heavily depends on training data. It can predict motion for different objects at the same positions, but is not able to adapt to objects at different positions. The small amount of training data is an important problem impairing the generalization of the trained models to different bowl positions. The experimental results tell us that a small amount of training data leads to bad results. A large amount of training data shows good prediction.

VI. CONCLUSIONS AND FUTURE WORK

This paper presented a bilateral teleoperation system for task learning and robotic motion generation. It trained DCAE and LSTM-RNN to learn scooping motion using data collected by human demonstration on the bilateral teleoperation system. The results showed the data collected using the bilateral teleoperation system was suitable for training deep learning models. The trained model was able to predict scooping motion for different objects at the same positions, showing some ability of generalization. The results also showed that the amount of data was an important issue that affect training good deep learning models.

One way to improve performance is to increase training data. However, increasing training data is not trivial for LfD applications since they require human operators to repeatedly

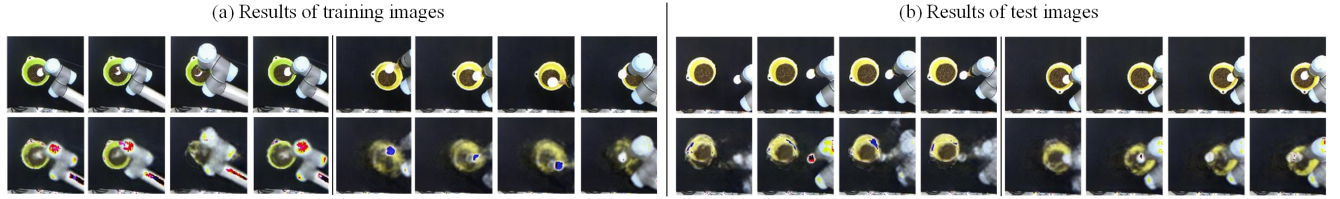


Fig. 10: The results of DCAE for experiment 1 and 2. (a.1-2) are the results of training data and test data for experiment 1. (b.1-2) are the results of training data and test data for experiment 2.

TABLE I: Results of scooping using DCAE and LSTM-RNN

	A1-A7r_s1	A1-A7r_s1s4	A1-A7r_s4	A1_s1	A4_s4	A1A4_s1s4
position s1	○	○	-	○	-	×
position s4	-	×	×	-	○	×

work on teaching devices. Another method is to use a mixed learning and planning model. Practitioners may use planning to collect data and use learning to generalize the planned results. The mixed method is our future direction.

ACKNOWLEDGMENT

The paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

REFERENCES

- [1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.
- [2] 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.
- [3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.
- [4] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.
- [5] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
- [6] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.
- [7] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems*, 2007, pp. 153–160.
- [8] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.
- [9] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.
- [10] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2009.
- [11] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.
- [12] J. Dequaire, P. Ondruška, D. Rao, D. Wang, and I. Posner, "Deep tracking in the wild: End-to-end tracking using recurrent neural networks," *The International Journal of Robotics Research*, pp. 492–512, 2017.
- [13] W. Wan, H. Liu, L. Wang, G. Shi, and W. J. Li, "A hybrid hmm/svm classifier for motion recognition using μ imu data," in *IEEE International Conference on Robotics and Biomimetics*, 2007, pp. 115–120.
- [14] 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, p. e1000220, 2008.
- [15] H. Arie, T. Endo, T. Arakaki, S. Sugano, and J. Tani, "Creating novel goal-directed actions at criticality: A neuro-robotic experiment," *New Mathematics and Natural Computation*, vol. 5, no. 01, pp. 307–334, 2009.
- [16] S. Jeong, H. Arie, M. Lee, and J. Tani, "Neuro-robotics study on integrative learning of proactive visual attention and motor behaviors," *Cognitive Neurodynamics*, vol. 6, no. 1, pp. 43–59, 2012.
- [17] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.
- [18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [19] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017.
- [20] Z. Yu, D. S. Moirangthem, and M. Lee, "Continuous timescale long-short term memory neural network for human intent understanding," *Frontiers in Neurobotics*, vol. 11, p. 42, 2017.
- [21] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," *arXiv preprint arXiv:1506.02078*, 2015.
- [22] H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll, and J. Schmidhuber, "A system for robotic heart surgery that learns to tie knots using recurrent neural networks," *Advanced Robotics*, vol. 22, no. 13-14, pp. 1521–1537, 2008.
- [23] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," *arXiv preprint arXiv:1707.02920*, 2017.
- [24] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, "Imitation from observation: Learning to imitate behaviors from raw video via context translation," *arXiv preprint arXiv:1707.03374*, 2017.
- [25] F. Abi-Farraj, N. Pedemonte, and P. R. Giordano, "A visual-based shared control architecture for remote telemanipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 4266–4273.
- [26] P. F. Hokayem and M. W. Spong, "Bilateral teleoperation: An historical survey," *Automatica*, vol. 42, no. 12, pp. 2035–2057, 2006.