IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 25, 2022, accepted February 11, 2022, date of publication February 28, 2022, date of current version March 30, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3155255

An Independently Learnable Hierarchical Model for Bilateral Control-Based Imitation Learning Applications

KAZUKI HAYASHI[1], **SHO SAKAINO**[2], **(Member, IEEE), AND TOSHIAKI TSUJI**[3], **(Senior Member, IEEE)**

[1]Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba 305-8577, Japan
[2]Department of Intelligent Interaction Technologies, University of Tsukuba, Tsukuba 305-8577, Japan
[3]Department of Electrical and Electronic Systems, Saitama University, Saitama 338-8570, Japan

Corresponding author: Kazuki Hayashi (s2020780@s.tsukuba.acjp)

This work was supported in part by the Adaptable and Seamless Technology Transfer Program through the Target-Driven Research and Development (A-STEP) from the Japan Science and Technology Agency (JST) under Grant JPMJTR20RG, and in part by the Japan Society for the Promotion of Science by the Grant-in-Aid for Scientific Research (B) under Grant 21H01347.

ABSTRACT Recently, motion generation by machine learning has been actively researched to automate various tasks. Imitation learning is one such method that learns motions from data collected in advance. However, executing long-term tasks remains challenging. Therefore, a novel framework for imitation learning is proposed to solve this problem. The proposed framework comprises upper and lower layers, where the upper layer model, whose timescale is long, and lower layer model, whose timescale is short, can be independently trained. In this model, the upper layer learns long-term task planning, and the lower layer learns motion primitives. The proposed method was experimentally compared to hierarchical RNN-based methods to validate its effectiveness. Consequently, the proposed method showed a success rate equal to or greater than that of conventional methods. In addition, the proposed method required less than 1/20 of the training time compared to conventional methods. Moreover, it succeeded in executing unlearned tasks by reusing the trained lower layer.

INDEX TERMS Bilateral control, imitation learning, motion planning, robot learning.

I. INTRODUCTION

Recently, motion generation based on machine learning has been studied to automate various processes using robots. For robotic automation, two methods, namely, reinforcement learning and imitation learning, have been mainly researched.

In the first method, the robots learn motions autonomously by repeating trials. However, this method needs many trials to acquire the motion policies [1]. To solve this problem, Sim2real, in which robots repeat trials in a simulation environment instead of the real world, was proposed [2]. Although this method is effective, robots sometimes cannot behave properly in practice because of the gap between the simulated and real environments [3].

On the other hand, in imitation learning, robots acquire motion from expert data collected in advance [4], [5]. Therefore, this requires significantly fewer demonstrations than that of reinforcement learning. Subsequently, imitation learning is practical for generating motions in robots. Moreover, imitation learning with force information enables robots to perform various tasks with contacts [6]–[10]. In particular, neural networks (NNs) are promising techniques for inferring complicated motions. Recurrent NNs (RNNs), which enable time-series inference, are effective for generating robotic motions [11]. However, imitation learning still has problems when executing long-term tasks [12]–[14].

Long short-term memory (LSTM) has been proposed to improve the long-term inference of RNNs but its performance remains insufficient. In addition, when multiple sensors are used simultaneously, the different sampling periods of the sensors can be a problem. To address this problem, two methods for dividing a large task into multiple subtasks have been studied.

The first approach treats many subtasks as a monolithic task. This method does not require subtask segmentation. Although RNN has limitations with respect to the length of

The associate editor coordinating the review of this manuscript and approving it for publication was Cheng Chin.

the time series, it can be solved by introducing a hierarchical structure [15], [16]. Saito *et al.* succeeded in executing tasks, such as wiping on a complex surface with a multi-timescale RNN (MTRNN) [17]. MTRNN is a type of hierarchical RNN that consists of three nodes that update every time constant. However, to consider longer tasks and operations in various environments, an enormous number of demonstrations are needed [18].

The second approach is to divide a long-term task into several subtasks or primitives. After segmentation, it is possible to execute various tasks by rearranging or re-utilizing them. This method needs fewer demonstrations compared to the first approach. However, the segmentation of the task sometimes needs human intervention [19]. Although many methods for automating this process have been researched [20]–[25], most of them require strong mathematical assumptions or prior knowledge about the task [26].

Considering that the behavior of a mechanical system is governed by position and force, we can infer that the segmentation of tasks will be easier if the position and force information can be accurately reproduced. In fact, past research has shown that extracting movement primitives by utilizing the interaction force results in better generalization abilities [27]. Subsequently, we proposed bilateral control-based imitation learning that enables robots to execute tasks requiring force adjustment and fast behavior [28]–[32]. Bilateral control is a remote-control technique for leader and follower robots with force feedback [31]. The force information collected with bilateral control is helpful for learning the movement primitive. Therefore, we propose a hierarchical imitation learning method for bilateral control-based imitation learning, which has the merits of both abovementioned approaches. In other words, our method does not require explicit task segmentation, instead few demonstrations are required.

In our proposed method, two types of LSTM, an upper layer LSTM and a lower layer LSTM, are used. In the proposed model, the inference of the slow-moving higher layer can be regarded as almost unchanged from perspective of the fast-moving lower layer. In the proposed method, the upper layer LSTM is trained to learn long-term task planning from the time series of the follower state. Then, the lower layer LSTM is trained to learn motion primitives from the states of the leader and follower robots. In the training stage of the lower layer LSTM, the force information brought by bilateral control improves the generalization ability and adaptability to environmental changes. Note that conventional hierarchical imitation learning cannot accurately reproduce position and force responses in the frequency domain; thus, it is necessary to maintain a mechanism where the upper and lower layers feedback each other to correct errors. However, in our bilateral control-based imitation learning framework because both position and force responses can be accurately reproduced in the frequency domain, the feedback from the lower layer to the upper layer is unnecessary. Therefore, the upper and lower layers can be trained independently. The features of our proposed method are summarized as follows.

1) The upper layer can learn long-term task planning without task segmentation.
2) The lower layer LSTM learns motion primitives by utilizing the force information collected by bilateral control.
3) The upper layer LSTM and lower layer LSTM are independently trained. This feature reduces training time. Additionally, the lower layer infers motion primitives and allows robots to perform various tasks.
4) The proposed method retains characteristics of bilateral control-based imitation learning, such as the fast movement, force adjustment, and adaptivity to environmental changes.
5) It enables robots to execute longer-term tasks.

In this study, the proposed method was compared to three conventional hierarchical RNN types: MTRNN [33], Clockwork RNN [34], and Fast-Slow RNN [35]. In the experiments, our proposed method showed a performance equal to or better than that of the conventional methods. Moreover, training with our method required less than 1/20 of the training time compared to conventional methods. In addition, the lower LSTM can be reused for an unlearned task by writing unlearned characters.

The remainder of this paper is structured as follows. Section II introduces the procedure of bilateral control-based imitation learning. This section consists of data collection (demonstration) with bilateral control, preprocessing, and training with the collected data. In section III, our proposed method and conventional hierarchical RNNs are described. In section IV, the experimental procedures and results are explained. Here, the types of conventional hierarchical RNNs are compared to our proposed method. In section V, the conclusion of this study and future research directions are described.

II. RELATED WORK

In imitation learning, robots acquire skills from demonstrations. On the other hand, bilateral control is the teleoperation method between two robots, the leader manipulated by an expert and the follower who operates in the workspace. During the bilateral control, angles and torques of the follower and the leader are synchronized. Moreover, collecting demonstrations with bilateral control has several advantages.

Firstly, it enables robots to operate fast. As described above, the state of the follower and that of the leader are synchronized. In other words, the command value for the follower is the state of the leader. Thus, robots can operate quickly by using the predicted leader's state as the command value. On the other hand, this is impossible when bilateral control is not used because the appropriate command value is unavailable. In addition, this trait helps variable speed motion generation [36].

Second, it allows the robot to operate with the right adjustment of force. In bilateral control, the torque responses of the follower and the leader are synchronized and the law of action and reaction is established between them. Here, the leader measures the action force, and the follower observes the reaction force. Thus, collecting demonstrations with bilateral control enables robots to imitate the force adjustments [30].

However, in the past, our model tended to be unstable in autonomous operation because it was trained with teacher forcing [31]. To solve this problem, Sasagawa *et al.* proposed the FL2FL model to train models with scheduled sampling. Although only the follower performs tasks in autonomous operation, the FL2FL needs two inputs, the states of the follower and the leader. Thus, the predicted leader state and the current follower's state are used as the inputs in using the FL2FL model [31]. However, this procedure causes the covariate shift and destabilizes the autonomous operation. Hence the F2FL model was proposed to solve it [32].

III. BILATERAL CONTROL-BASED IMITATION LEARNING

In general imitation learning approaches, such as direct teaching, only one robot's responses are available and next step responses are treated as commands. However, because the commands were substituted for the responses, only low-frequency operations could be realized if responses and commands could be assumed to be consistent. Contrarily, commands for a follower are responses of a leader during bilateral control. Therefore, both commands and responses for a follower are available in bilateral control-based imitation learning [31]. Therefore, our bilateral control-based imitation learning enables robots to operate quickly owing to using the predicted leader state as a command value. Moreover, the force information is obtained as the torque response in our method. Therefore, our method is suitable for executing many tasks that require force information and adaptivity to environmental changes.

A. DATA COLLECTION WITH BILATERAL CONTROL

4ch bilateral control was used to collect training datasets [30]. In this study, experts manipulate the leader, while the follower is teleoperated by the leader. This is obtained by the synchronizing the positions and feedback of the forces between the leader and follower robots. The control goals of the bilateral control are summarized as follows:

$$\theta_l^{res} - \theta_f^{res} = 0 \tag{1}$$

$$\tau_l^{res} + \tau_f^{res} = 0. \tag{2}$$

Here, the superscripts *res* represent the response value, and l and f indicate the leader and follower, respectively. Moreover, θ and τ denote the angle and torque, respectively. During data collection, the control period of the robots was 1 ms.

The details can be found in our previous research [31], [32]. Note that the leader is referenced as the master, and the follower is called the slave in these studies.

B. PREPROCESSING

After collecting the motion data, the sampling rate of the training data was decimated to 20 ms, and the rejected data were reused for data augmentation [37]. Afterward, min-max normalization was executed on the decimated data. Here, the states of the follower and leader in each batch at the t-th time step are defined as follows:

$$L_t = [\theta_l(t), \dot{\theta}_l(t), \tau_l(t)] \tag{3}$$

$$F_t = [\theta_f(t), \dot{\theta}_f(t), \tau_f(t)]. \tag{4}$$

C. F2L MODEL

This is the basic model for bilateral control-based imitation learning, which is the same as the S2M model described in [31].

1) TRAINING

The model is trained to predict the next leader's response value L_{t+1} from the current follower's response value F_t, as depicted in Fig. 1.

2) AUTONOMOUS OPERATION

As presented in Fig. 2, the trained model receives the current follower's state and predicts the next leader's state, which is used as the command value for the follower during its autonomous operation. Although the F2L model succeeded in some tasks, it is not suitable for executing long-term tasks. This is because the F2L model does not consider the accumulation of prediction errors during autonomous operations.

D. FL2FL MODEL (FOR AUTOREGRESSIVE LEARNING)

This model is the same as the SM2SM model described in [31]. To solve the problem of the F2L model, the FL2FL model was proposed.

1) TRAINING

This model is trained to predict the next states of the follower and leader $[F_{t+1}, L_{t+1}]$ from the current states of the follower and leader $[F_t, L_t]$. In addition, the predicted states of the follower and leader $[\hat{F}_{t+1}, \hat{L}_{t+1}]$ are used as the model input in the next step, as depicted in Fig. 3. This is called autoregressive learning or scheduled sampling, and Sasagawa *et al.* showed that it is effective for generating motions for long-term tasks [31].

2) AUTONOMOUS OPERATION

During autonomous operations, the leader's response value does not exist because only the follower robot operates alone. Therefore, the predicted state of the leader in the previous step and the current state of the follower were used as the model input, as depicted in Fig. 4. Similar to the F2L model, the predicted value of the leader's response was treated as the command for the follower. However, using the predicted leader state as the model input sometimes causes a covariate shift [38].

E. F2FL MODEL (FOR AUTOREGRESSIVE LEARNING)

This model is the same as the S2SM model described in [32] and is regarded as an improved type of the FL2FL model. To solve the covariate shift problem of the FL2FL model, the

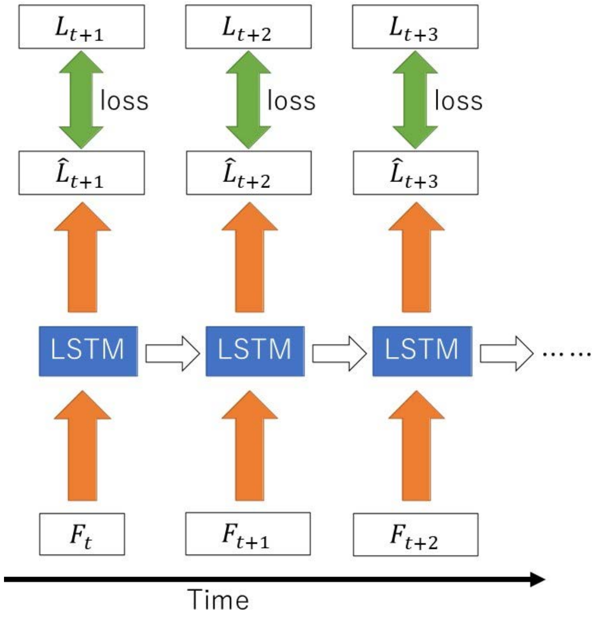

FIGURE 1. Training of the F2L model.

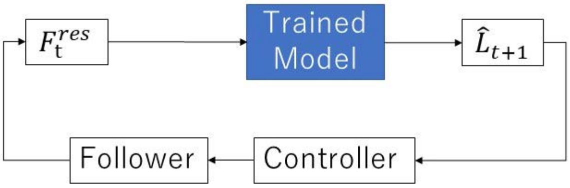

FIGURE 2. Autonomous operation using the F2L model.

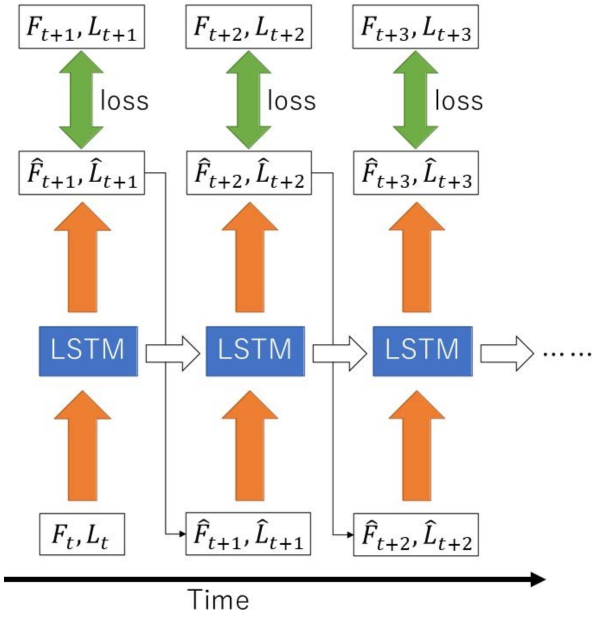
F_{t+1}, L_{t+1}
F_{t+2}, L_{t+2}
F_{t+3}, L_{t+3}
loss
loss
loss
$\hat{F}_{t+1}, \hat{L}_{t+1}$
$\hat{F}_{t+2}, \hat{L}_{t+2}$
$\hat{F}_{t+3}, \hat{L}_{t+3}$
LSTM
LSTM
LSTM
……
F_t, L_t
$\hat{F}_{t+1}, \hat{L}_{t+1}$
$\hat{F}_{t+2}, \hat{L}_{t+2}$
Time

FIGURE 3. Training of the FL2FL model. Note that the predicted states of the follower and leader $\hat{L}_t, \hat{F}_t$ are used as the input in the next step.

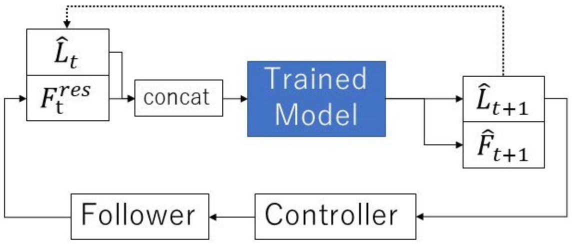

FIGURE 4. Autonomous operation using the FL2FL model. Note that the leader state predicted previously is used as the next input because only the follower operates.

F2FL model proposes eliminating the leader's responses from the inputs.

1) TRAINING

The F2FL model is trained to predict the next states of the follower and leader $[F_{t+1}, L_{t+1}]$ from the current state of the follower F_t. Additionally, the predicted state of the follower $\hat{F}_{t+1}$ is used as the next input of the model, as presented in Fig. 5.

2) AUTONOMOUS OPERATION

This model receives the current state of the follower and predicts the next states of the follower and leader. Moreover, the predicted response of the leader is used as the command value for the follower, as depicted in Fig. 6. Note that the responses of the leader are not required to reduce the covariate shit effect.

F. DETAILED EXPLANATION OF AUTOREGRESSIVE LEARNING

1) INTRODUCTION OF SCHEDULED SAMPLING

As explained in section III-C, the model input is the follower state at time step t (F_t), and its output is the next leader state (L_{t+1}) used in training the F2L model. Note that the model input is always the follower state of the training data. This training method is called "teacher forcing" because the model input is always the values of the teacher (training) data. Ranzato *et al.* asserted that training with teacher forcing decreases robustness because the model never trains on its own error [39]. To solve this problem, scheduled sampling was proposed [40]. In training models with scheduled sampling, the model output at the previous step is used as the input, as explained in sections III-D and III-E.

2) QUALIFY THE AUTOREGRESSIVE LEARNING'S EFFECTIVENESS

In summary, the model trained with scheduled sampling is expected to generate appropriate behavior even if the input is slightly different from the probabilistic distribution of the training data. Hence, we studied whether the generated motion was similar to the training data by comparing their probability distributions.

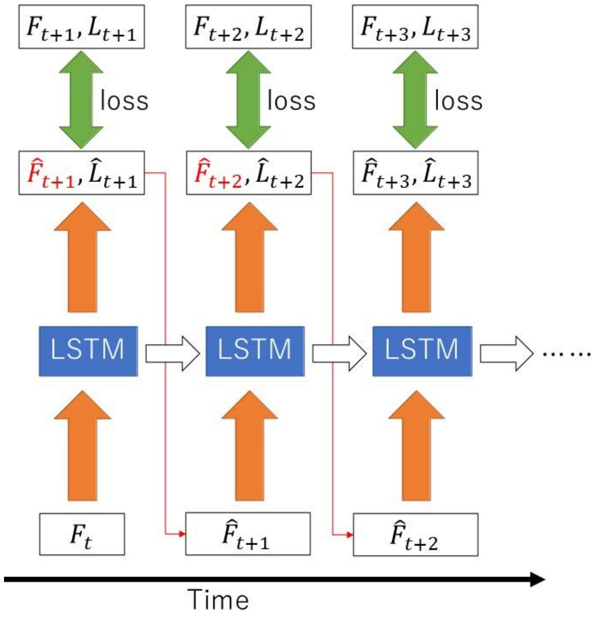

FIGURE 5. Training of the F2FL model. Note that only the predicted follower state is used as the input in the next step.

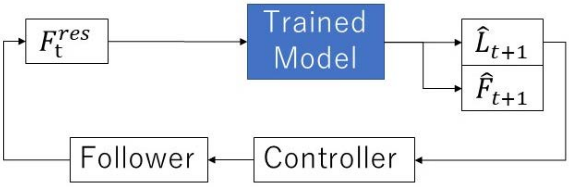

FIGURE 6. Autonomous operation using the F2FL model.

The Pearson (PE) divergence, which measures the difference between two probabilistic distributions $p(Y), p'(Y)$, is defined as follows:

$$PE(P||P') := \frac{1}{2}\int p'(Y)\left(\frac{p(Y)}{p'(Y)} - 1\right)^2 dY. \quad (5)$$

According to [41], the similarity is defined as follows:

$$PE(P_t||P_{t+n}) + PE(P_{t+n}||P_t). \quad (6)$$

Here, P_t is the distribution of samples in the Hankel matrix Y_t, which was obtained from the time-series $[y_1, \ldots, y_T]$. This index refers to the difference in the distribution of a single time series at different time steps and is used for change-point detection. Therefore, let us define P'_t as the distribution of samples in the Hankel matrix X_t, which was obtained from the time-series $[x_1, \ldots, x_T]$.

Moreover, the similarity of the distributions of two time-series data $[x_1, \ldots, x_T]$, $[y_1, \ldots, y_T]$ in the same time step t is defined as follows:

$$PE(P_t||P'_t) + PE(P'_t||P_t). \quad (7)$$

We used this index to qualify the effectiveness of autoregressive learning.

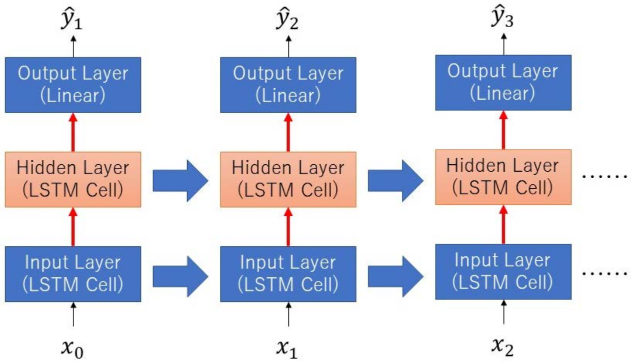

FIGURE 7. Diagram of our basic LSTM when the number of hidden layers is one. It consists of multiple LSTM cells with the same node size and a fully connected layer.

IV. TYPES OF NEURAL NETWORKS FOR HIERARCHICAL IMITATION LEARNING

A. BASIC LSTM STRUCTURE

As depicted in Fig. 7, our basic LSTM consists of an input layer (LSTM cell), a hidden layer (LSTM cell), and an output layer (fully connected layer). Moreover, each LSTM cell had the same node size. Therefore, all hyperparameters were the node size and the number of layers.

B. PROPOSED METHOD I (ORDINARY MODEL)

In the proposed method I, the upper and lower layers are trained independently.

1) TRAINING OF THE UPPER LAYER LSTM

The upper LSTM is trained to predict the state of the follower 20 steps later F_{t+20} from the current follower's state F_t. In training, note that the predicted follower state $\hat{F}_{t+20}$ is used as the input in the next step, as presented in Fig. 8 (autoregressive learning). As well as in section IV-A, the LSTM model with the configuration shown in Fig. 7 was used.

2) TRAINING OF THE LOWER LAYER LSTM

The lower layer LSTM is trained to predict the next states of the follower and leader $[\hat{F}_{t+1}, \hat{L}_{t+1}]$ from the current state of the follower F_t (and the leader state L_t) and the future follower's state F_{t+20}, as shown in Fig. 9. Note that the future follower state F_{t+20} is updated every 20 steps. In addition to the training of the FL2FL, the predicted states $[\hat{F}_{t+1}, \hat{L}_{t+1}]$ are used as the input in the next step. In summary, the lower layer is the F2FL or FL2FL model that uses F_{t+20} as its input. As well as in section IV-A, the LSTM model with the configuration shown in Fig. 7 was used.

3) AUTONOMOUS OPERATION

Fig. 10 demonstrates the procedure for autonomous operation with the follower. In autonomous operation, the upper layer receives the follower response F_t^{res} and outputs its future goal state $\hat{F}_{t+20}$. Then, the lower layer predicts the next states of the follower and leader $[\hat{F}_{t+1}, \hat{L}_{t+1}]$ from the follower response F_t and its future goal state $\hat{F}_{t+20}$. Similar to other models, the predicted leader response at the next time step $\hat{L}_{t+1}$ is used as the command value. Moreover, the future goal state $\hat{F}_{t+20}$ is updated every 20 time steps.

C. PROPOSED METHOD II (ANGLE MODEL)

1) TRAINING OF THE UPPER LAYER LSTM

The upper layer LSTM is trained to predict the future angles of the four followers $[\theta_f(t+10), \theta_f(t+20), \theta_f(t+30), \theta_f(t+40)]$ from the current follower angle $\theta_f(t)$, as shown in Fig. 11. Furthermore, the predicted angle $\hat{\theta}_f(t+10)$ is used as the next input (autoregressive learning). As well as in section IV-A, the LSTM model with the configuration shown in Fig. 7 was used.

2) TRAINING OF THE LOWER LAYER LSTM

The lower layer LSTM is trained to predict the next states of the follower and leader $[\hat{F}_{t+1}, \hat{L}_{t+1}]$ from the current state of the follower F_t (and the leader state L_t) and the future follower angles $[\theta_f(t+10), \theta_f(t+20), \theta_f(t+30), \theta_f(t+40)]$, as depicted in Fig. 12. Note that the future follower angles are updated every 10 steps. In addition to the training of the FL2FL, the predicted states $[\hat{F}_{t+1}, \hat{L}_{t+1}]$ are used as the input in the next step. In summary, the lower layer is the F2FL or FL2FL model that uses the future goal angles $[\theta_f(t+10), \theta_f(t+20), \theta_f(t+30), \theta_f(t+40)]$ as its input. As well as in section IV-A, the LSTM model with the configuration shown in Fig. 7 was used.

3) AUTONOMOUS OPERATION

Fig. 13 demonstrates the procedure for autonomous operation with the follower. In autonomous operation, the upper layer receives the follower's angular response $\theta_f^{res}(t)$ and outputs its future goal angles $[\hat{\theta}_f(t+10), \hat{\theta}_f(t+20), \hat{\theta}_f(t+30), \hat{\theta}_f(t+40)]$.

Then, the lower layer predicts the next states of the follower and leader $[\hat{F}_{t+1}, \hat{L}_{t+1}]$ from the follower's response F_t^{res} and its future goal angles. Similar to other models, the predicted leader response at the next time step $\hat{L}_{t+1}$ is used as the command value. Moreover, the future goal angles $[\hat{\theta}_f(t+10), \hat{\theta}_f(t+20), \hat{\theta}_f(t+30), \hat{\theta}_f(t+40)]$ are updated every 10 time steps.

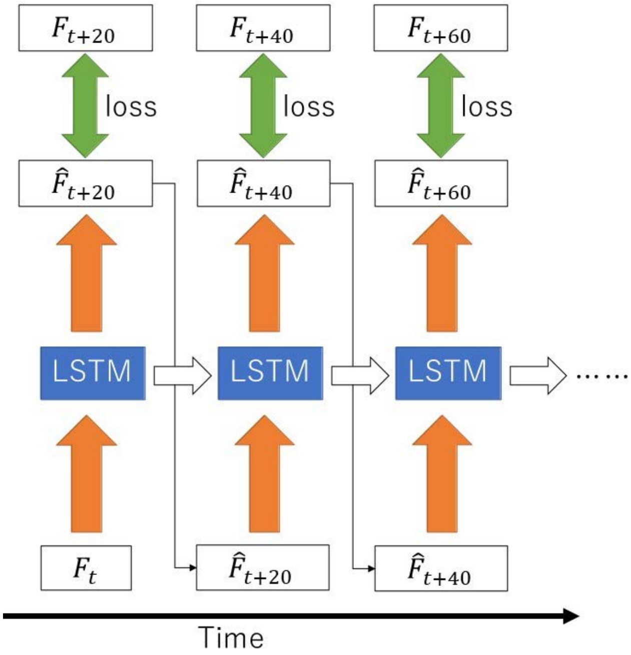

FIGURE 8. Training of the upper layer of the proposed method I. The timescale of the upper layer is 20 steps in the proposed method I.

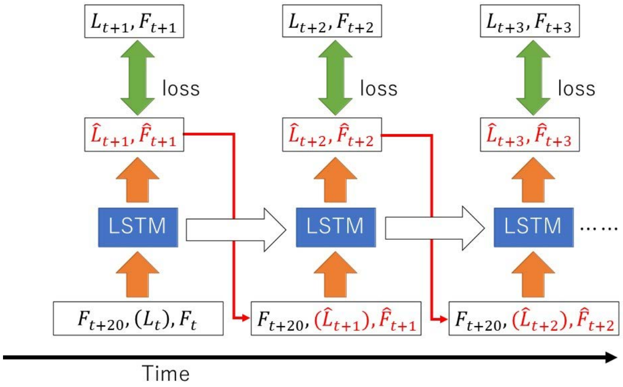
L_{t+1}, F_{t+1}
L_{t+2}, F_{t+2}
L_{t+3}, F_{t+3}
loss
loss
loss
$\hat{L}_{t+1}, \hat{F}_{t+1}$
$\hat{L}_{t+2}, \hat{F}_{t+2}$
$\hat{L}_{t+3}, \hat{F}_{t+3}$
LSTM
LSTM
LSTM
……
$F_{t+20}, (L_t), F_t$
$F_{t+20}, (\hat{L}_{t+1}), \hat{F}_{t+1}$
$F_{t+20}, (\hat{L}_{t+2}), \hat{F}_{t+2}$
Time

FIGURE 9. Training of the lower layer of the proposed method I. Note that the future follower state F_{20} is updated every 20 time steps.

D. COMPARISON METHODS

1) FAST-SLOW RNN (FS-RNN)

Fast-Slow RNN is a hierarchical RNN type [35]. As presented in Fig. 14, it consists of two types of RNN cells, namely, fast and slow cells. During the forward calculation of FS-RNN, the fast RNN cell sends its hidden state to the slow RNN cell once every few steps. When the slow RNN cell receives the hidden state, the slow RNN cell returns the hidden state to the fast RNN cell. Because of this procedure, the fast RNN cell learns the short-term components, while the slow RNN cell learns the long-term dependencies. In our experiments, two fully connected (FC) layers were attached to the slow cell, as shown in Fig. 14. This is because short-term information that fast cell stores are required to control robots. The node size was the only hyperparameter because the node sizes of layers should be equal. Moreover, the state of the upper layer was updated every 20 steps as well as the proposed method I.

2) CLOCKWORK RNN (CW-RNN)

Clockwork RNN is a hierarchical RNN type as well [34]. As depicted in Fig. 15, the hidden units are divided into modules in the Clockwork RNN. Moreover, the i-th module is updated every 2^i-th time-step. Owing to this structure, Clockwork RNN is computationally efficient and can learn

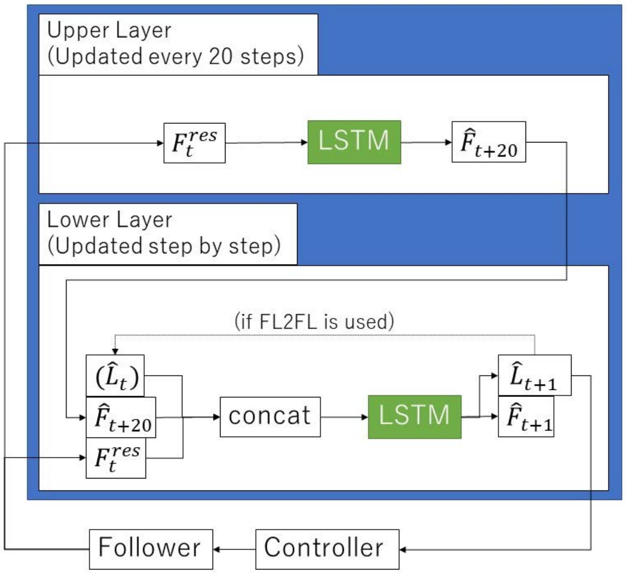

FIGURE 10. Autonomous operation with the proposed method I. Note that the future follower state $\hat{F}_{t+20}$ is updated every 20 time steps.

long-term dependencies. Moreover, hyperparameters were the node size and the number of divided modules.

3) MULTIPLE TIMESCALE RECURRENT NEURAL NETWORK (MTRNN)

As presented in Fig. 16, MTRNN is a type of hierarchical RNN that is structured by three types of nodes with different time constants [17], [33]. These are slow context (Cs) nodes, fast context (Cf) nodes, and input/output (I/O) nodes. The Cs nodes have a longer time constant and are expected to learn the long-term dependencies. On the other hand, Cf nodes can learn motion primitives because they have a shorter time constant. Owing to these three types of nodes, MTRNN enables robots to perform long-term tasks.

In the forward calculation, the state of the *i*-th neuron at time step *t* is calculated as follows:

$$u_i(t) = \sigma\left\{\left(1 - \frac{1}{\tau_i}\right) u_i(t-1) + \frac{1}{\tau_i}\left(\sum_{j \in N} w_{ij} x_j(t)\right)\right\}. \quad (8)$$

Here, N denotes the number of neurons connected to neuron i. Moreover, w_{ij} is the weight from neuron j to neuron i, and σ denotes the sigmoid activation function. Then, the output value is calculated with the sigmoid function as follows:

$$y_i(t) \ = \sigma(u_{io}(t)), \quad (9)$$

where, u_{io} is the neuron at I/O nodes. As shown in Fig.16, i-th neuron and N neurons connected to it are described in Table 1.

Moreover, hyperparameters were the node sizes and time constants of the Cf and Cs nodes.

TABLE 1. Neurons that are connected to the *i*-th neuron.

neuron i	N(neurons connected to neuron i)
I/O	Fast
Fast	I/O, Fast, Slow
Slow	Fast, Slow

TABLE 2. The specifications of the computer that was used for the training and autonomous operation.

OS	Ubuntu 18.04.6 LTS (64bit)
RAM	32 GB
CPU	AMD Ryzen 7 3700x 8-core procesor x 16
GPU	NVIDIA GeForce RTX 2080 SUPER/PCIe/SSE2

V. EXPERIMENT

In the experiment, four types of tasks were executed to demonstrate the effectiveness of the proposed method. The proposed method was compared to CW-RNN, FS-RNN, MTRNN, and LSTM. Here, LSTM denotes our conventional method, which has no hierarchical structure [30]–[32]. During the experiments, the FL2FL and F2FL models were used inside the comparison methods, CW-RNN, FS-RNN, MTRNN, and LSTM. Moreover, the F2L and FL2FL models were used in the MTRNN because the model input and output sizes should be equal when using the MTRNN. In this study, three autoregression numbers 1, 5, 20 were used. These numbers indicate the frequency of scheduled sampling. For example, Fig. 5 demonstrates the training case with an autoregression number of 3. In summary, we consider three steps of time-series data as one block of in the training. In other words, the value obtained from the training data is used once every three times.

Table 2 lists the specifications of the computer that was used for the training and autonomous operation.

A. EXPERIMENT 1 (WRITING THE CHARACTER B)

1) TASK DESIGN

In this experiment, a ballpoint pen was fixed to the follower, as depicted in Fig. 17. The task is to write the letter ‘B’ without any mistakes. The purpose of this experiment is to reveal the characteristics of each method. For example, it remains unknown how many autoregressions are appropriate for each comparative method. It is difficult to provide an answer directly because the more the autoregressive number increases, the more training time is needed. Hence, it is required to know model characteristics and utilize them for subsequent experiments.

2) DATA COLLECTION & TRAINING

In total, 18 motion data were collected with bilateral control at three paper heights: 70, 45, and 20 mm. The task duration was approximately 6.5 seconds (325 time-steps) and that of each data was 28.02 seconds (1401 time-steps). Thus, each dataset contains four demonstrations. In training, the parameters are determined by Bayesian optimization with optuna [42], as presented in Table 3.

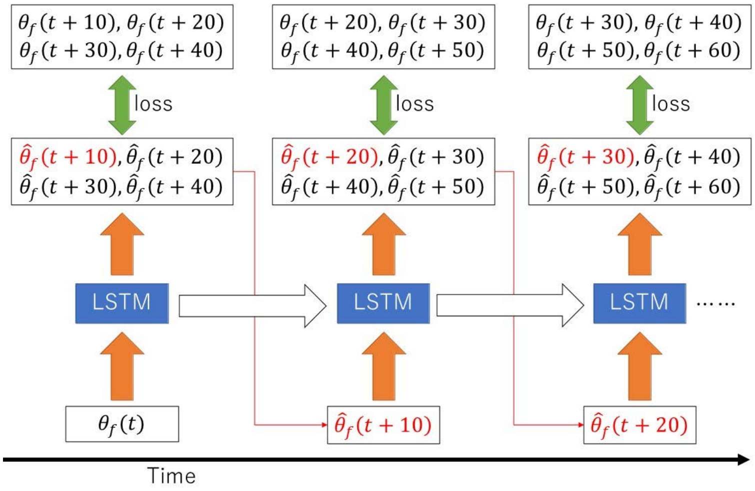
$\theta_f(t+10), \theta_f(t+20)$
$\theta_f(t+30), \theta_f(t+40)$
$\theta_f(t+20), \theta_f(t+30)$
$\theta_f(t+40), \theta_f(t+50)$
$\theta_f(t+30), \theta_f(t+40)$
$\theta_f(t+50), \theta_f(t+60)$
loss
loss
loss
$\hat{\theta}_f(t+10), \hat{\theta}_f(t+20)$
$\hat{\theta}_f(t+30), \hat{\theta}_f(t+40)$
$\hat{\theta}_f(t+20), \hat{\theta}_f(t+30)$
$\hat{\theta}_f(t+40), \hat{\theta}_f(t+50)$
$\hat{\theta}_f(t+30), \hat{\theta}_f(t+40)$
$\hat{\theta}_f(t+50), \hat{\theta}_f(t+60)$
LSTM
LSTM
LSTM
……
$\theta_f(t)$
$\hat{\theta}_f(t+10)$
$\hat{\theta}_f(t+20)$
Time

FIGURE 11. Training of the upper layer of the proposed method II. The timescale of the upper layer is 10 steps in the proposed method II.

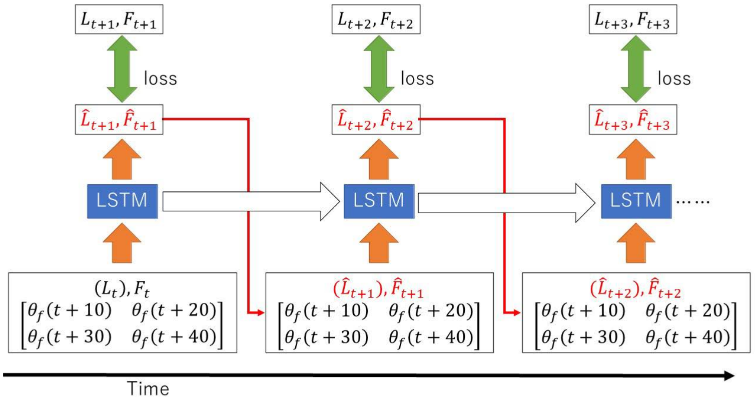
L_{t+1}, F_{t+1}
L_{t+2}, F_{t+2}
L_{t+3}, F_{t+3}
loss
loss
loss
$\hat{L}_{t+1}, \hat{F}_{t+1}$
$\hat{L}_{t+2}, \hat{F}_{t+2}$
$\hat{L}_{t+3}, \hat{F}_{t+3}$
LSTM
LSTM
LSTM
……
$(L_t), F_t$
$\begin{bmatrix}\theta_f(t+10) & \theta_f(t+20)\\ \theta_f(t+30) & \theta_f(t+40)\end{bmatrix}$
$(\hat{L}_{t+1}), \hat{F}_{t+1}$
$\begin{bmatrix}\theta_f(t+10) & \theta_f(t+20)\\ \theta_f(t+30) & \theta_f(t+40)\end{bmatrix}$
$(\hat{L}_{t+2}), \hat{F}_{t+2}$
$\begin{bmatrix}\theta_f(t+10) & \theta_f(t+20)\\ \theta_f(t+30) & \theta_f(t+40)\end{bmatrix}$
Time

FIGURE 12. Training of the lower layer of the proposed method II. Note that the future follower state $\theta_f(t+10), \theta_f(t+20), \theta_f(t+30), \theta_f(t+40)$ is updated every 10 time steps.

3) TASK VALIDATION

After the training of each model, the autonomous operation was executed at five paper heights of 70, 60, 45, 35, and 20 mm. Note that the paper heights 60 and 35 mm were not trained. The standard of the task's success is determined considering whether the follower succeeded in writing the letter 'B' without any mistakes five times sequentially.

4) EXPERIMENT RESULTS

As presented in Table 4, autonomous operation tends to be unsuccessful when the autoregressive number is low. This result is as predicted because previous research has shown that autoregressive learning is effective in improving the performance of autonomous operations [31], [32]. Therefore, the autoregressive number was fixed at 20 in the comparative methods during subsequent experiments. However, the training of the FL2FL model with MTRNN could not be completed within 24 h. Thus, its autoregressive number was fixed at 1.

5) EVALUATION OF THE AUTOREGRESSIVE LEARNING WITH THE PE DIVERGENCE

In the previous section, the effect of the autoregressive learning was qualitatively given. Then, its quantitative performance is analyzed here by comparing the PE divergence of the LSTM model (our conventional model) with and without the autoregressive learning. In this analysis, the PE divergence between the autonomous operation and training data was computed. In Fig. 18, the follower response value during the autonomous operation and state in the training data are plotted. In addition, a Hankel matrix was created from this time series, and the PE divergence was calculated as depicted in Fig. 19. As a result, the PE divergence was high in the case of operation without autoregressive learning. This result proves that autoregressive learning reduces the

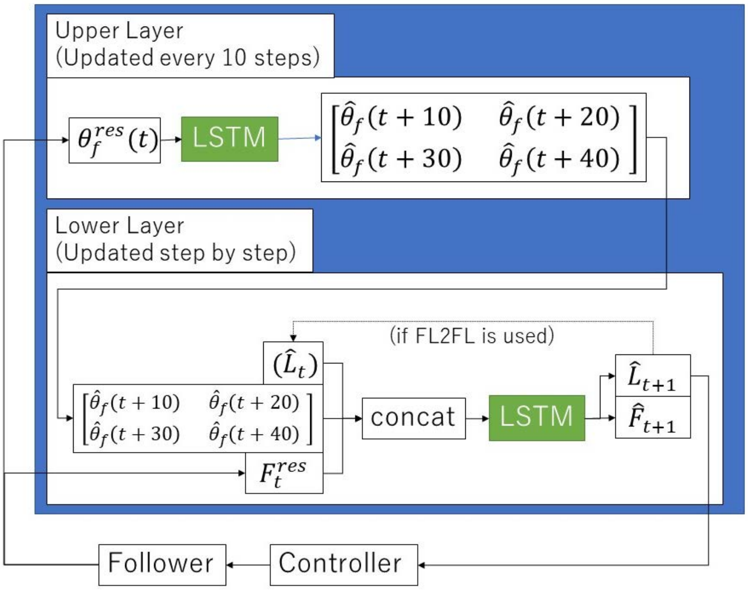
Upper Layer
(Updated every 10 steps)
$\theta_f^{res}(t)$
LSTM
$\begin{bmatrix}\hat{\theta}_f(t+10) & \hat{\theta}_f(t+20)\\ \hat{\theta}_f(t+30) & \hat{\theta}_f(t+40)\end{bmatrix}$
Lower Layer
(Updated step by step)
(if FL2FL is used)
$(\hat{L}_t)$
$\begin{bmatrix}\hat{\theta}_f(t+10) & \hat{\theta}_f(t+20)\\ \hat{\theta}_f(t+30) & \hat{\theta}_f(t+40)\end{bmatrix}$
F_t^{res}
concat
LSTM
$\hat{L}_{t+1}$
$\hat{F}_{t+1}$
Follower
Controller

FIGURE 13. Autonomous operation using proposed method II. Note that the future follower state $[\hat{\theta}_f(t+10), \hat{\theta}_f(t+20), \hat{\theta}_f(t+30), \hat{\theta}_f(t+40)]$ is updated every 10 time steps.

TABLE 3. Hyper parameters gained by optuna in experiment 1. These were calculated from the bayesian optimization by optuna. In the column of the CW-RNN, "Node" represents the node size, and "module" means the number of divisions. Also in other columns," Node" means the node size. In the column of MTRNN, "τ" means the time constant. In the columns of LSTM, "LSTM Cells" represents the number of the stacked LSTM cells. A more detailed explanation of these hyperparameters can be found in the section IV.

CW-RNN		FS-RNN		MTRNN		LSTM	
F2FL	FL2FL	F2FL	FL2FL	F2L	FL2FL	F2FL	FL2FL
module:5 Node:400	module:5 Node:400	Node:650	Node:650	Node_fast:550 Node_slow:150 τ_fast:10 τ_slow:350	Node_fast:450 Node_slow:250 τ_fast:40 τ_slow:100	Node:200 LSTM Cells:2	Node:180 LSTM Cells:1

TABLE 4. Result of experiment 1 (Writing the letter B). Each check mark means success at all five heights. Because the training of the FL2FL model with MTRNN was not finished within 24 hours, their columns say "None."

	CW-RNN		FS-RNN		MTRNN		LSTM	
Autoregressive number	F2FL	FL2FL	F2FL	FL2FL	F2L	FL2FL	F2FL	FL2FL
1 (w/o autoregression)	-	-	-	-	✓	-	-	-
5	-	-	-	-		None	-	-
20	✓	✓	✓	-		None	✓	✓

influence of the covariate shift and helps robots perform tasks successfully.

B. EXPERIMENT 2 (WRITING THE CHARACTERS ABC)

1) TASK DESIGN

The goal of this task was to write three letters, "A," "B," and "C" sequentially without any mistakes. The execution of this task takes approximately 20 s (1000 steps). In addition, adaptation to changes in height is necessary. Therefore, the robot needs to be able to both plan long-term tasks and adjust the force properly.

2) DATA COLLECTION & TRAINING

The motion data were collected with bilateral control. Here, demonstrations were conducted at paper heights of 70 mm, 45 mm, and 20 mm. A total of 30 data points were collected. Moreover, the duration of each data is 60.4 s (3020 time-steps) and that of the task is approximately 20 s (1000 time-steps). Thus, each dataset contained three demonstrations. As shown in Fig. 20, experts wrote along the printed letters. In training, hyperparameters are determined according to a Bayesian optimization with optuna [42], as reported in Table 5 and 6.

3) TASK VALIDATION

After training each model, the autonomous operation was executed at five paper heights of 70, 60, 45, 35, and 20 mm. Note that the paper heights 60 and 35 mm were not trained. First, we focused on the epoch and loss graphs and identified an epoch number E_{med} that could perform the task

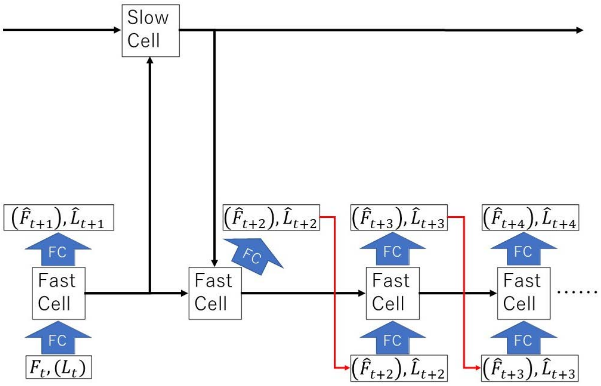
Slow Cell
Fast Cell
Fast Cell
Fast Cell
Fast Cell
......
FC
FC
FC
FC
FC
FC
FC
$(\hat{F}_{t+1}), \hat{L}_{t+1}$
$(\hat{F}_{t+2}), \hat{L}_{t+2}$
$(\hat{F}_{t+3}), \hat{L}_{t+3}$
$(\hat{F}_{t+4}), \hat{L}_{t+4}$
$F_t, (L_t)$
$(\hat{F}_{t+2}), \hat{L}_{t+2}$
$(\hat{F}_{t+3}), \hat{L}_{t+3}$

FIGURE 14. Training of the FL2FL or F2FL model with a fast-slow RNN. FC stands for a fully connected layer, and red arrows indicate the data flow of autoregressive learning adopted in the FL2FL or F2FL models.

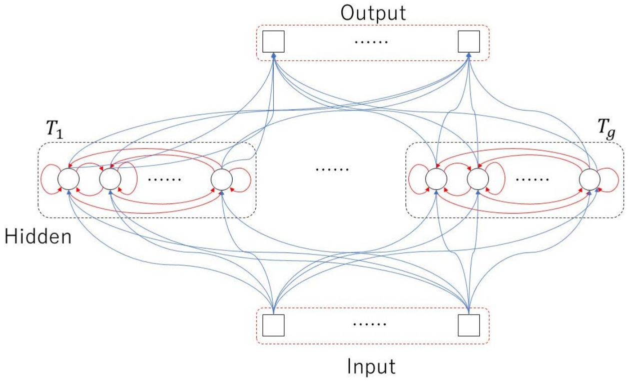

FIGURE 15. Diagram of a Clockwork RNN [34]. The hidden unit is divided into g modules and each module is updated in different cycles.

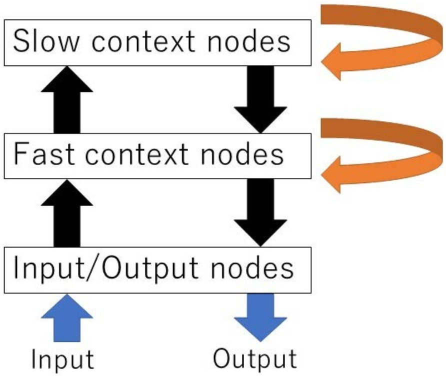

FIGURE 16. Diagram of a MTRNN. Each arrow indicates data flow among the nodes.

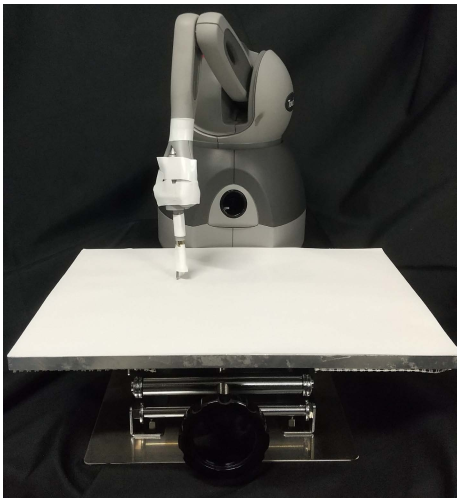

FIGURE 17. The settings of experiments 1, 2, and 3.

effectively. Subsequently, we performed autonomous operations with five models with epoch numbers of E_{med}, $E_{med} \pm 1000$, $E_{med} \pm 2000$, and calculated their success rates. In the autonomous operation using the proposed methods, this procedure was performed on the upper layer. During the training of the proposed methods, the autoregressive numbers of the upper and lower layers were 1, 5, and 20, respectively. As a result, the upper layer of the autoregressive number needed to be 5 or 20 to successfully execute the tasks. However, the lower layer did not require autoregression. Therefore, the autoregressive number of the upper layer was five and that of the lower layer was one in this experiment because small autoregressive numbers are preferable to shorten the training time.

4) EXPERIMENTAL RESULT

Table 7 reports the success rates of the experiment. In the row ‘‘Success or Failure’’, ‘‘Success’’ denotes that a model that can execute tasks at every height was found. ‘‘Partial Success’’ denotes that a model that can execute tasks at limited heights was found. ‘‘Failure’’ denotes that a model that can execute tasks was not found. Our proposed methods and CW-RNN and MTRNN succeeded in performing tasks at every height. Among these methods, the second proposed method type exhibited a higher success rate and reproducible trajectories, as depicted in Fig. 21 and Table 7. During autonomous operation, a misalignment of the fixed ballpoint pen sometimes occurs. This phenomenon had a negative influence on torque information. However, its influence was limited in the second proposed method because the upper layer only included positional (angular) information.

On the other hand, the first proposed method required the least time to train, as reported in Table 8. Both proposed methods require 1/20 less training time compared to the other methods. This is because their upper layer timescales were large, and their input and output shapes were symmetric.

C. EXPERIMENT 3 (WRITING UNLEARNED CHARACTERS)

1) TASK DESIGN

In this experiment, the most notable advantage of our proposed method was proven. The task was to write three letters, ‘‘X,’’ ‘‘Y,’’ and ‘‘Z’’ sequentially without any mistakes. In our methods, the upper and lower layers were

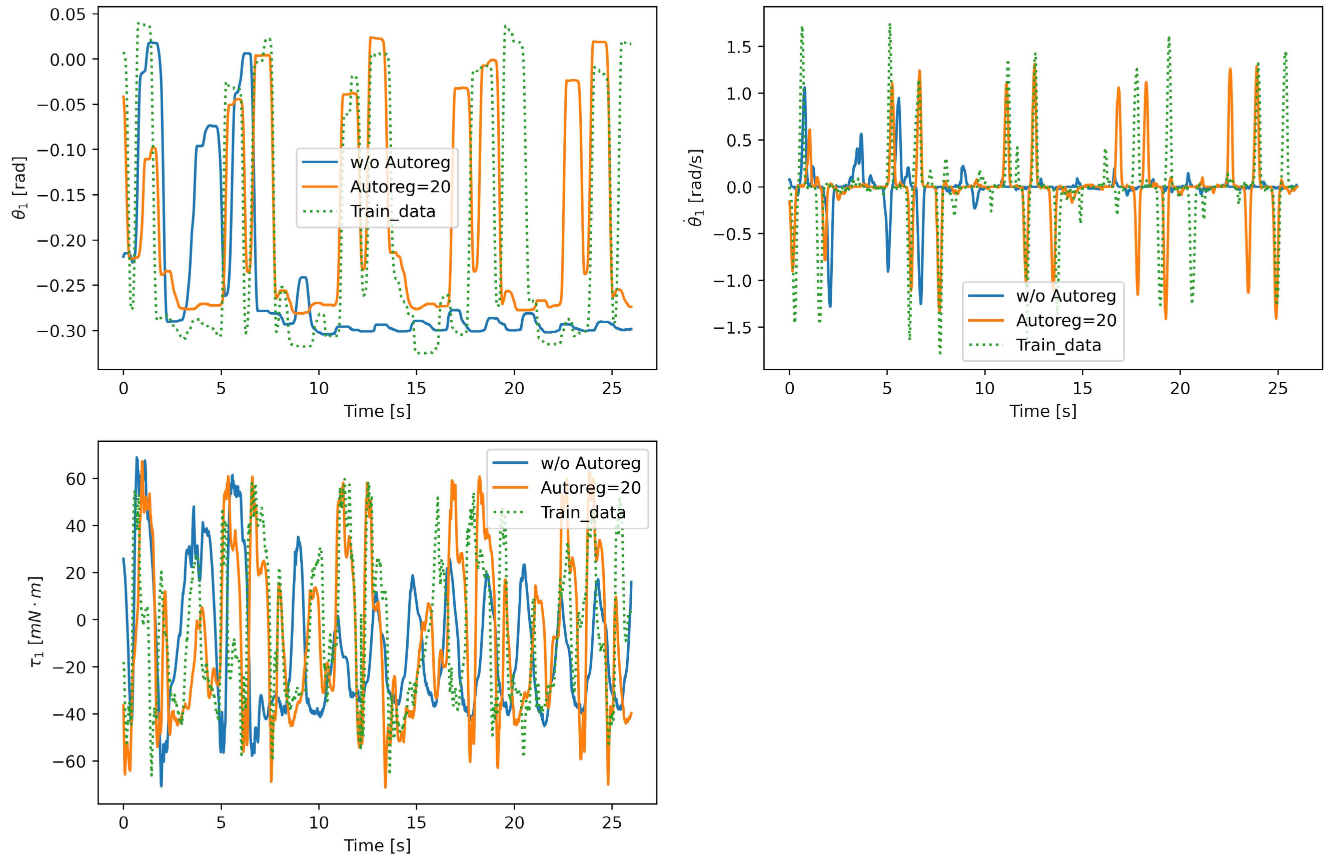

FIGURE 18. Autonomous operation with LSTM and training data in experiment 1 when the paper height was 20 mm. From the left, the first angle, angular velocity, and torque are shown.

TABLE 5. Hyper parameters of the proposed methods in experiment 2. These hyperparameters were calculated from the bayesian optimization by optuna. "Node" represents the node size, and "LSTM Cells" represents the number of the stacked LSTM cells. A more detailed explanation of these hyperparameters can be found in the section IV.

	Prop. 1		Prop. 2	
	F2FL	FL2FL	F2FL	FL2FL
Upper	Node:120 LSTM Cells:1		Node:170 LSTM Cells:1	
Lower	Node:100 LSTM Cells:1	Node:120 LSTM Cells:1	Node:170 LSTM Cells:1	Node:90 LSTM Cells:1

independently trained. During the training, the upper layer learns the long-term task plan, while the lower layer learns the motion primitive. Hence, various untrained tasks can be executed by utilizing the trained lower layer. Furthermore, this approach improves computational efficiency.

Especially in proposed method II, the upper layer includes only angle information and not force information. The positional (angular) trajectories can be computed from the kinematics or obtained by direct teaching. In another way, the higher layer can be trained using reinforcement-learning-based methods, such as sim2real [43]. Therefore, rough positional trajectories can be created using such methods, and the positional, velocity, and force commands can be generated by the lower layer. Using these methods, various tasks can be executed. In this experiment, we used the lower layer of the proposed method II trained in experiment 2, and the collected motion data were input to the $[\hat{\theta}_{t+10}, \hat{\theta}_{t+20}, \hat{\theta}_{t+30}, \hat{\theta}_{t+40}]$ parameters of the lower layer.

2) DATA COLLECTION

In the data collection phase, the follower robot was manipulated like a direct teaching approach, and the angular response of the follower robot was recorded. At this time, data collection was conducted to write along frames printed on the paper, as presented in Fig. 22. Note that bilateral control was not used, and training was not conducted here.

3) TASK VALIDATION

Subsequently, the recorded follower angular responses were input to the $[\hat{\theta}_{t+10}, \hat{\theta}_{t+20}, \hat{\theta}_{t+30}, \hat{\theta}_{t+40}]$ parameters of the lower layer of the second proposed method.

4) EXPERIMENTAL RESULTS

As depicted in Fig. 23, the outline shape was successfully written by exchanging the upper layer for the angular

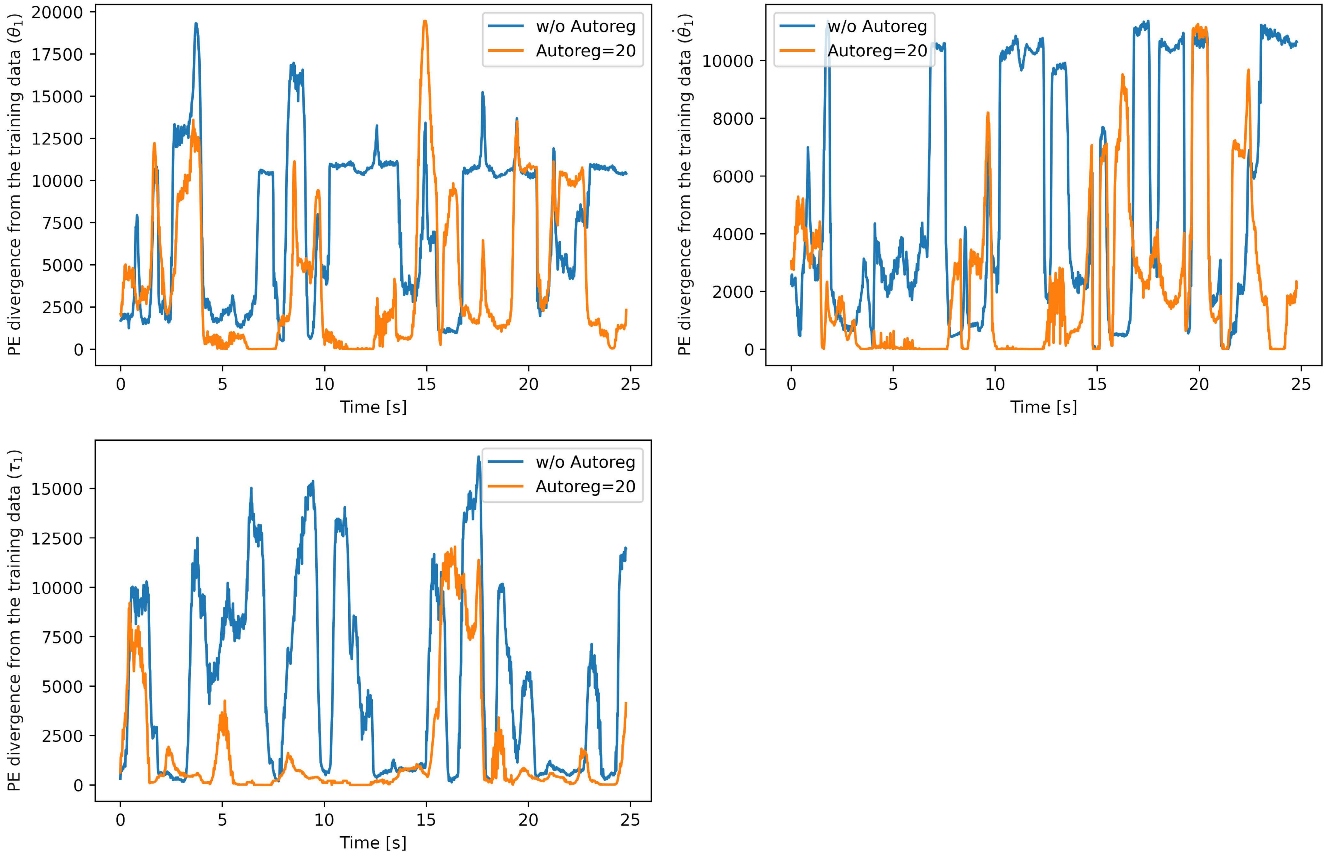

FIGURE 19. PE divergence between autonomous operation with LSTM and training data in experiment 1 when the paper height was 20 mm. From the left, the first angle, angular velocity, and torque are shown.

TABLE 6. Hyper parameters of the comparative methods in experiment 2. These were calculated from the bayesian optimization by optuna. In the column of the CW-RNN, "Node" represents the node size, and "module" means the number of divisions. Also in other columns," Node" means the node size. In the column of MTRNN, "τ" means the time constant. In the columns of LSTM, "LSTM Cells" represents the number of the stacked LSTM cells. A more detailed explanation of these hyperparameters can be found in the section IV.

CW-RNN		FS-RNN		MTRNN		LSTM	
F2FL	FL2FL	F2FL	FL2FL	F2L	FL2FL	F2FL	FL2FL
module:4 Node:480	module:5 Node:450	Node:550	Node:650	Node_fast:550 Node_slow:250 τ_fast:10 τ_slow:150	Node_fast:500 Node_slow:400 τ_fast:190 τ_slow:250	Node:110 LSTM Cells:1	Node:70 LSTM Cells:1

TABLE 7. The autonomous operation success rate of every model in experiment 2.

Paper	Prop. 1		**Prop. 2**		CW-RNN		FS-RNN		MTRNN		LSTM	
height [mm]	F2FL	FL2FL	**F2FL**	FL2FL	F2FL	FL2FL	F2FL	FL2FL	F2L	FL2FL	F2FL	FL2FL
70	100%	-	**100%**	100%	40%	-	80%	-	40%	-	-	-
60	40%	-	**100%**	0%	40%	-	40%	-	40%	-	-	-
45	100%	-	**100%**	100%	40%	-	60%	-	60%	-	-	-
35	40%	-	**100%**	100%	80%	-	20%	-	80%	-	-	-
20	100%	-	**100%**	100%	80%	-	20%	-	80%	-	-	-
Success or Failure	Success	Failure	**Success**	Partial Success	Success	Failure	Partial Success	Failure	Success	Failure	Failure	Failure

trajectory collected in advance. This result indicates that the trained lower layer appropriately generated angle, angular velocity and torque commands only from angular responses and is helpful for executing various unlearned tasks. However, its shape was slightly different from that of the recorded follower state because of the limited diversity of training data.

TABLE 8. Training time of each model in experiment 2 (writing ABC).

		Prop. 1		Prop. 2		CW-RNN		FS-RNN		MTRNN		LSTM	
		F2FL	FL2FL	F2FL	FL2FL	F2FL	FL2FL	F2FL	FL2FL	F2L	FL2FL	F2FL	FL2FL
Upper	Epoch	**40000**	-	26000	-								
	Time[s]	**1300**	-	1815	-								
Lower	Epoch	**2400**	-	1000	-	28000	-	-	-	92000	-	-	-
	Time[s]	**898**	-	183	-	37816	-	-	-	71024	-	-	-

TABLE 9. Hyper parameters of the proposed method in experiment 4. These hyperparameters were calculated from the bayesian optimization by optuna. "Node" represents the node size, and "LSTM Cells" represents the number of the stacked LSTM cells. A more detailed explanation of these hyperparameters can be found in the section IV.

	Prop. 1		Prop. 2	
	F2FL	FL2FL	F2FL	FL2FL
Upper	Node:120 LSTM Cells:1		Node:170 LSTM Cells:1	
Lower	Node:100 LSTM Cells:1	Node:160 LSTM Cells:1	Node:170 LSTM Cells:1	Node:170 LSTM Cells:1

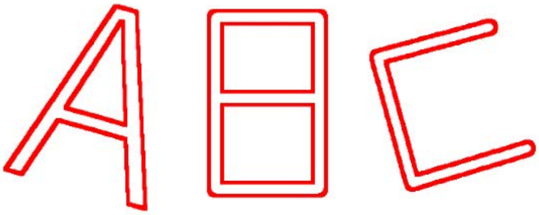

FIGURE 20. The letters printed on the paper for experiment 2.

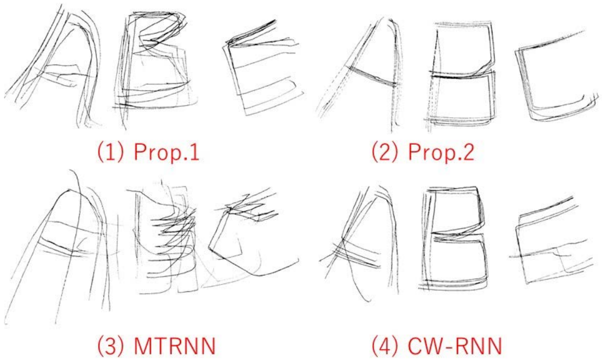

FIGURE 21. Letters written by each method.

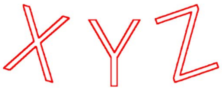

FIGURE 22. The letters printed on the paper for experiment 3.

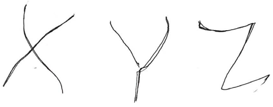

FIGURE 23. Letters written in experiment 3.

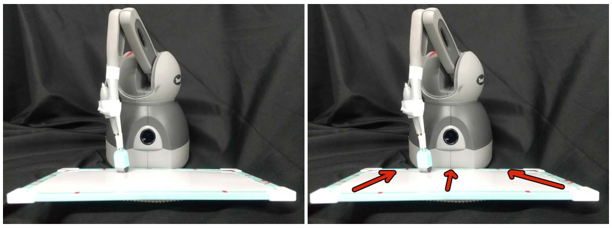

FIGURE 24. The left figure presents the setting of experiment 4, and the right figure shows three erasing directions.

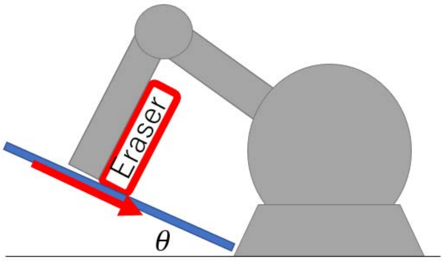

FIGURE 25. The side view of the setting of experiment 4.

D. EXPERIMENT 4 (WIPING THE WHITEBOARD WITH THE ERASER)

1) TASK DESIGN

Figs. 24 and 25 present the setting of this experiment. In this experiment, the whiteboard eraser was fixed to the follower robot. The task was to wipe the tilted whiteboard in three different directions. Executing this task approximately required 30 s (1500 steps). Additionally, the follower robot was required to adapt to the changes in the tilt angle of the whiteboard. Therefore, the robot needed to be able to both plan long-term tasks and adjust the force properly. Moreover, note that the fixed whiteboard eraser periodically moved from side to side while wiping. This means that our approach is also effective for other tasks, such as grinding off parts.

2) DATA COLLECTION & TRAINING

In this task, a robot wiped the tilted plate in three directions. Here, the robot rubbed the plate with a fixed eraser, similar to using an eraser manually. In total, 9 data were collected on a tilted plate with bilateral control. In these demonstrations, the tilt angles of the plate were 0°, 10°, and 20°. Furthermore, the data duration was 60.40 s (3020 timesteps) and that of the

TABLE 10. Hyper parameters of the comparative methods in experiment 4. These were calculated from the bayesian optimization by optuna. In the column of the CW-RNN, "Node" represents the node size, and "module" means the number of divisions. Also in other columns," Node" means the node size. In the column of MTRNN, "τ" means the time constant. In the columns of LSTM, "LSTM Cells" represents the number of the stacked LSTM cells. A more detailed explanation of these hyperparameters can be found in the section IV.

CW-RNN		FS-RNN		MTRNN		LSTM	
F2FL	FL2FL	F2FL	FL2FL	F2L	FL2FL	F2FL	FL2FL
module:4 Node:360	module:3 Node:270	Node:850	Node:1150	Node_fast:550 Node_slow:50 τ_fast:10 τ_slow:300	Node_fast:450 Node_slow:550 τ_fast:10 τ_slow:200	Node:200 LSTM Cells:2	Node:130 LSTM Cells:1

TABLE 11. The autonomous operation success rate of every model in experiment 4.

Degree of	**Prop. 1**		Prop. 2		CW-RNN		FS-RNN		**MTRNN**		LSTM	
a tilted plate [deg]	**F2FL**	FL2FL	F2FL	FL2FL	F2FL	FL2FL	F2FL	FL2FL	**S2M**	FL2FL	F2FL	FL2FL
20	**100%**	-	100%	0%	40%	-	-	-	**100%**	-	-	-
15	**100%**	-	100%	0%	60%	-	-	-	**80%**	-	-	-
10	**80%**	-	80%	20%	0%	-	-	-	**100%**	-	-	-
5	**80%**	-	80%	0%	40%	-	-	-	**80%**	-	-	-
0	**100%**	-	60%	80%	80%	-	-	-	**100%**	-	-	-
Success or Failure	**Success**	Failure	Success	Parital Success	Partial Success	Failure	Failure	Failure	**Success**	Failure	Failure	Failure

TABLE 12. Training time required by each model in experiment 4.

		Prop. 1		Prop. 2		CW-RNN		FS-RNN		MTRNN		LSTM	
		F2FL	FL2FL	F2FL	FL2FL	F2FL	FL2FL	F2FL	FL2FL	F2L	FL2FL	F2FL	FL2FL
Upper	Epoch	**39000**	-	26000	-								
Layer	Time[s]	**1344**	-	3616	-								
Lower	Epoch	**9000**	-	3000	-	-	-	-	-	49000	-	-	-
Layer	Time[s]	**3380**	-	544	-	-	-	-	-	75324	-	-	-

task was approximately 30 s (1500 time-steps). Therefore, each data point contains two demonstrations. In addition to the second experiment, in training, the hyperparameters were determined by Bayesian optimization with optuna [42], as reported in Tables 9 and 10.

3) TASK VALIDATION

After training each model, the autonomous operation was executed at five tilt angles of: 20°, 15°, 10°, 5°, and 0°. Note that the tilt angles 15° and 5° were not trained. First, we focused on the epoch and loss graphs and identified an epoch number E_{med} that could perform the task. Subsequently, we performed autonomous operations with five models with epoch numbers of E_{med}, $E_{med} \pm 1000$, $E_{med} \pm 2000$, and calculated the model success rates. In the autonomous operation with the proposed methods, this procedure was performed on the upper layer. When training the proposed methods, the autoregressive numbers of the upper and lower layers were 1, 5, and 20, respectively. As a result, the upper and lower layer autoregression numbers needed to be 5 or 20 to execute the tasks successfully. Therefore, the autoregression number of both layers was fixed at five. In this task, a success criterion is whether a robot can wipe in all three directions without mistakes.

4) EXPERIMENTAL RESULT

The success rate of the experiment is reported in Table 11. In the row "Success or Failure," "Success" denotes that a model that can execute tasks at every tilt angle was found. "Partial Success" denotes that a model that can execute tasks at certain tilt angles was found. "Failure" denotes that a model that can execute tasks was not found. Our proposed methods and MTRNN succeeded in performing tasks at every angle, as reported in Table 12. In these methods, the first proposed method type exhibited a higher success rate. This is because the upper layer of the first proposed method includes the force information, which helps the robot recognize the current situation in a long-term task. In addition, both proposed methods required 1/20 of the training time required by the comparative methods.

VI. DISCUSSION

Although the ordinary LSTM could succeed in only the first experiment, the proposed method could execute subsequent tasks. These results demonstrate the effectiveness of the hierarchical structure. In the third experiment, the trained lower layer succeeded in writing unlearned characters by receiving positional trajectories collected with direct teaching. However, performance improvement is expected when the diversity of the training datasets is increased in the data collection phase. In another method, data augmentation is considered. In the experiments, the proposed method required less than 1/20 of the training time compared to conventional methods. In the comparative methods, more exponential training time is required for tasks that are more complex and require long-term inference. However, in the proposed method, the exponential increase in training time can be suppressed by using a structure with three or more levels of hierarchy. Furthermore, we have succeeded in integrating the motion inference and images taken with a the camera, which works at a slow sampling rate. Hence, the proposed method can adapt to complex tasks that require multiple sensors with different sampling frequencies. However, detailed information is not described here because it is beyond the scope of this research. In addition, because the upper and lower layers are trained independently, they can be designed using different NN models. For example, a model with an upper layer provided by reinforcement learning and a lower layer characterized where by LSTM may be useful. With this combination, the upper layer, which is trained with sim2real, generates various trajectories, and lower layer treats force information. In the future, we will apply the proposed method to longer-term tasks by adding three or more levels of hierarchy to ensure that more complex tasks can be executed by reusing multiple trained lower layers.

VII. CONCLUSION

As an expansion of our past research, a bilateral control-based hierarchical imitation learning framework was proposed here. In our proposed method, the upper layer, whose timescale is large, and lower layer, whose timescale is short, are separately trained. With this framework, robots can execute more long-term tasks while maintaining the advantages of bilateral control-based imitation learning, such as fast movement and force adjustment.

In experiment 2, the first proposed method required less than 1/20 of the training time compared to other hierarchical methods. Additionally, the second proposed method demonstrated the best reproducibility. In experiment 3, the lower layer of the proposed method, which was trained in experiment 2, succeeded in writing unlearned characters. This is because the lower layer learns the motion primitive and receives a rough positional trajectory.

In addition, in experiment 4, the proposed method required less than 1/20 of the training time compared to other hierarchical methods. Moreover, although this task is the most difficult among all experiments because it requires 30 s, the proposed method exhibited the best success rate here.

As described above, the proposed method can be used for a difficult task that requires long-term inference, dynamic motion generation, and control of the contact force with the environment simultaneously. Another feature of the proposed method is that it can be trained with a very short training time. Moreover, because it does not use inductive bias for a task, it is likely to be useful for general tasks. In the future, we will apply our method to perform various tasks and verify its general usefulness.

REFERENCES

[1] 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. Robot. Res.*, vol. 37, nos. 4–5, pp. 421–436, Apr. 2018.

[2] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 23–30.

[3] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," 2021, *arXiv:2109.11978*.

[4] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, and F. Sun, "Survey of imitation learning for robotic manipulation," *Int. J. Intell. Robot. Appl.*, vol. 3, no. 4, pp. 362–369, 2019.

[5] A. Hussein, M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, p. 35, 2017.

[6] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, no. 2, p. 17, Apr. 2018.

[7] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Adv. Robot.*, vol. 25, no. 5, pp. 581–603, 2011.

[8] A. X. Lee, H. Lu, A. Gupta, S. Levine, and P. Abbeel, "Learning force-based manipulation of deformable objects from multiple demonstrations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 177–184.

[9] M. Edmonds, F. Gao, X. Xie, H. Liu, S. Qi, Y. Zhu, B. Rothrock, and S.-C. Zhu, "Feeling the force: Integrating force and pose for fluent discovery through imitation learning to open medicine bottles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 3530–3537.

[10] L. Rozo, P. Jiménez, and C. Torras, "A robot learning from demonstration framework to perform force-based manipulation tasks," *Intell. Service Robot.*, vol. 6, no. 1, pp. 33–51, 2013.

[11] K. Nguyen, "Imitation learning with recurrent neural networks," 2016, *arXiv:1607.05241*.

[12] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 3, no. 1, pp. 297–330, May 2020.

[13] H. Zhang, Y. Liu, and W. Zhou, "Long time sequential task learning from unstructured demonstrations," *IEEE Access*, vol. 7, pp. 96240–96252, 2019.

[14] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Proc. Conf. Robot Learn.*, 2020, pp. 1025–1037.

[15] S. E. Hihi and Y. Bengio, "Hierarchical recurrent neural networks for long-term dependencies," in *Proc. Adv. Neural Inf. Process. Syst.*, 1995, pp. 493–499.

[16] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," in *Proc. 5th Int. Conf. Learn. Represent.*, 2019.

[17] N. Saito, D. Wang, T. Ogata, H. Mori, and S. Sugano, "Wiping 3D-objects using deep learning model based on image/force/joint information," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10152–10157.

[18] T. Yu, P. Abbeel, S. Levine, and C. Finn, "One-shot hierarchical imitation learning of compound visuomotor tasks," 2018, *arXiv:1810.11043*.

[19] S. Lee and S. Seo, "Learning compound tasks without task-specific knowledge via imitation and self-supervised learning," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, 2020, pp. 5747–5756.

[20] Y. Wu and Y. Demiris, "Hierarchical learning approach for one-shot action imitation in humanoid robots," in *Proc. 11th Int. Conf. Control Automat. Robot. Vis.*, Dec. 2010, pp. 453–458.

[21] R. Fox, R. Berenstein, I. Stoica, and K. Goldberg, "Multi-task hierarchical imitation learning for home automation," in *Proc. IEEE 15th Int. Conf. Automat. Sci. Eng. (CASE)*, Aug. 2019, pp. 1–8.

[22] S. Niekum, "Complex task learning from unstructured demonstrations," in *Proc. AAAI*, 2021, vol. 26, no. 1, pp. 2402–2403.

[23] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. J. Lim, "Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1235–1245.

[24] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5239–5246.

[25] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *Int. J. Robot. Res.*, vol. 31, no. 3, pp. 360–375, Mar. 2012.

[26] S. Niekum, S. Chitta, B. Marthi, and S. Osentoski, "Incremental semantically grounded learning from demonstration," in *Robotics: Science and Systems*, vol. 9. 2013, pp. 15607–15610.

[27] J. Kober, M. Gienger, and J. J. Steil, “Learning movement primitives for force interaction tasks,” in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 3192–3199.

[28] K. Fujimoto, S. Sakaino, and T. Tsuji, “Time series motion generation considering long short-term motion,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 6842–6848.

[29] T. Adachi, K. Fujimoto, S. Sakaino, and T. Tsuji, “Imitation learning for object manipulation based on position/force information using bilateral control,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 3648–3653.

[30] A. Sasagawa, K. Fujimoto, S. Sakaino, and T. Tsuji, “Imitation learning based on bilateral control for human–robot cooperation,” *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6169–6176, Oct. 2020.

[31] A. Sasagawa, S. Sakaino, and T. Tsuji, “Motion generation using bilateral control-based imitation learning with autoregressive learning,” *IEEE Access*, vol. 9, pp. 20508–20520, 2021.

[32] K. Hayashi, A. Sasagawa, S. Sakaino, and T. Tsuji, “A new autoregressive neural network model with command compensation for imitation learning based on bilateral control,” in *Proc. IEEE Int. Conf. Mechatronics (ICM)*, Mar. 2021, pp. 1–7.

[33] Y. Yamashita and J. Tani, “Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment,” *PLoS Comput. Biol.*, vol. 4, no. 11, Nov. 2008, Art. no. e1000220.

[34] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, “A clockwork RNN,” in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1863–1871.

[35] A. Mujika, F. Meier, and A. Steger, “Fast-slow recurrent neural networks,” in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5917–5926.

[36] Y. Saigusa, A. Sasagawa, S. Sakaino, and T. Tsuji, “Imitation learning for variable speed motion generation over multiple actions,” in *Proc. 47th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2021, pp. 1–6.

[37] R. Rahmatizadeh, P. Abolghasemi, A. Behal, and L. Bölöni, “From virtual demonstration to real-world manipulation using LSTM and MDN,” 2016, *arXiv:1603.03833*.

[38] M. Sugiyama, M. Krauledat, and K. Müller, “Covariate shift adaptation by importance weighted cross validation,” *J. Mach. Learn. Res.*, vol. 812, no. 5, pp. 985–1005, 2017.

[39] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, “Sequence level training with recurrent neural networks,” 2015, *arXiv:1511.06732*.

[40] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, “Scheduled sampling for sequence prediction with recurrent neural networks,” in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, vol. 1, 2015, pp. 1171–1179.

[41] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, “Change-point detection in time-series data by relative density-ratio estimation,” *Neural Netw.*, vol. 43, pp. 72–83, Jul. 2013.

[42] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2623–2631.

[43] Y. Wang, C. C. Beltran-Hernandez, W. Wan, and K. Harada, “Robotic imitation of human assembly skills using hybrid trajectory and force learning,” 2021, *arXiv:2103.05912*.

KAZUKI HAYASHI received the B.E. degree in mechanical engineering from the Tokyo University of Agriculture and Technology, Tokyo, Japan, in 2020. He is currently pursuing the M.E. degree with the Department of Information and Systems, University of Tsukuba. His research interests include robotics, motion generation, and neural networks.

SHO SAKAINO (Member, IEEE) received the B.E. degree in system design engineering and the M.E. and Ph.D. degrees in integrated design engineering from Keio University, Yokohama, Japan, in 2006, 2008, and 2011, respectively. He was an Assistant Professor at Saitama University, from 2011 to 2019. Since 2019, he has been an Associate Professor at the University of Tsukuba. His research interests include mechatronics, motion control, robotics, and haptics. He has received the IEEJ Industry Application Society Distinguished Transaction Paper Award, in 2011 and 2020. He has also received the RSJ Advanced Robotics Excellent Paper Award, in 2020.

TOSHIAKI TSUJI (Senior Member, IEEE) received the B.E. degree in system design engineering and the M.E. and Ph.D. degrees in integrated design engineering from Keio University, Yokohama, Japan, in 2001, 2003, and 2006, respectively. He was a Research Associate with the Department of Mechanical Engineering, Tokyo University of Science, from 2006 to 2007. He is currently an Associate Professor with the Department of Electrical and Electronic Systems, Saitama University, Saitama, Japan. His research interests include motion control, haptics, and rehabilitation robots. He has received the FANUC FA and Robot Foundation Original Paper Award, in 2007 and 2008, respectively. He has also received the RSJ Advanced Robotics Excellent Paper Award and the IEEJ Industry Application Society Distinguished Transaction Paper Award, in 2020.