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

KAZUKI HAYASHI1, SHO SAKAINO2,(MEMBER, IEEE), AND TOSHIAKI TSUJI.3, (Senior Member, IEEE)

1Faculty of Engineering, Information and Systems, University of Tsukuba, Ibaraki 305-8577, Japan (e-mail: s2020780@s.tsukuba.ac.jp)
2Department of Intelligent Interaction Technologies, University of Tsukuba, Ibaraki 305-8577, Japan (e-mail: sakaino@iit.tsukuba.ac.jp)
3Department of Electrical and Electronic Systems, Saitama University, Saitama 338-8570, Japan (e-mail: tsuji@ees.saitama-u.ac.jp)

Corresponding author: Kazuki Hayashi (e-mail: s2020780@s.tsukuba.ac.jp).

This work was partly supported by the Adaptable and Seamless Technology Transfer Program through Target-driven R&D (A-STEP) from the Japan Science and Technology Agency (JST) Grant Number JPMJTR20RG.

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 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. 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:

\[
\begin{align*}
\dot{\theta}^\text{res}_f - \dot{\theta}^\text{res}_l &= 0 \quad (1) \\
\tau^\text{res}_l + \tau^\text{res}_f &= 0. \quad (2)
\end{align*}
\]

Here, the superscripts res represent the response value, and \( l \) and \( f \) indicate the leader and follower, respectively. Moreover, \( \theta \) and \( \tau \) 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 [36]. 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:

\[
L_t = [\theta(t), \dot{\theta}(t), \tau(t)]
\]

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

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 \( \hat{L}_{t+1} \) 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 \([\hat{F}_{t+1}, \hat{L}_{t+1}]\) 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 [37].

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 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 \([\hat{F}_{t+1}, \hat{L}_{t+1}] \) 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 shift effect.

F. DETAILED EXPLANATION OF AUTOREGRESSIVE LEARNING

1) Introduction of scheduled sampling

As explained in section II-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 [38]. To solve this problem, scheduled sampling was proposed [39]. In training models with scheduled sampling, the model output at the previous step is used as the input, as explained in sections II-D and II-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.

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 [40], 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.

III. 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 III-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 $[\hat{F}_{t+1}, L_{t+1}]$ 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 $[\hat{F}_{t+1}, L_{t+1}]$ 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 III-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 $\hat{F}_t$ 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 III-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 III-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 $\theta_{\text{res}}(t)$ 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 $[\hat{L}_{t+1}, \hat{L}_{t+1}]$ 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.

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 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:

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

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

$$y_i(t) = \sigma(u_{i,o}(t)),$$

where, $u_{i,o}$ 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.

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

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

**IV. 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)
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 processor x 16 |
| **GPU**    | NVIDIA GeForce RTX 2080 SUPER/PCIe/SSE2 |

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 [41], as presented in Table 3.

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 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 [41], 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
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 III.

|          | CW-RNN | FS-RNN | MTRNN | LSTM |
|----------|--------|--------|-------|------|
| module   | 5      | 5      | 5     | 5    |
| Node     | 400    | 650    | 650   | 200  |
| F2FL     | FL2FL  | FL2FL  | FL2L  | F2L  |
| Node_fast| 550    | 150    | 450   | 200  |
| Node_slow| 350    | 150    | 250   | 100  |
| τ_fast   | 10     | 40     | 10    |      |
| τ_slow   | 350    | 350    | 100   |      |
| τ         |        |        |       |      |

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".

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

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.

trained. First, we focused on the epoch and loss graphs and identified an epoch number $E_{med}$ that could perform the task 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 pro-
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.

The 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.

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

VOLUME 9, 2021
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 III.

| Upper | Lower |
|-------|-------|
| Node:120 LSTM Cells:1 | Node:100 LSTM Cells:1 |
| Node:170 LSTM Cells:1 | Node:170 LSTM Cells:1 |
| Node:90 LSTM Cells:1 | Node:90 LSTM Cells:1 |

TABLE 6. Hyper parameters of the comparative methods in experiment 2. These were calculated from the bayesian optimization by optuna. In the column of 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 III.

| CW-RNN | FS-RNN | MTRNN | LSTM |
|--------|--------|-------|------|
| module:4 Node:480 | module:5 Node:450 | Node:550 Node:650 | Node:fast:550 Node:slow:250 |
| Node:fast:500 Node:slow:400 | τ:fast:10 τ:slow:150 | Node:fast:900 Node: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 height [mm] | Prop. 1 | Prop. 2 | CW-RNN | FS-RNN | MTRNN | LSTM |
|-------------------|--------|--------|--------|--------|-------|------|
| 70                | 100%   | -      | 100%   | 100%   | 40%   | 80%  |
| 60                | 40%    | -      | 100%   | 0%     | 40%   | 40%  |
| Success or Failure| Success| Failure| Success| Partial Success| Success| Failure |
| 45                | 100%   | -      | 100%   | 100%   | 80%   | 20%  |
| 35                | 40%    | -      | 100%   | 100%   | 80%   | 20%  |
| 20                | 100%   | -      | 100%   | 100%   | 80%   | 80%  |
| Success or Failure| Success| Failure| Success| Partial Success| Failure| Success |

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

| Upper | Lower |
|-------|-------|
| Epoch | 40000 | 24000 |
| Time[s] | 1300 | 898 |
| Epoch | 26000 | 1815 |
| Time[s] | 1815 | 183 |

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 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 [42]. 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 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.

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 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 [41], 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.

V. 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 information 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.

VI. 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 handEye coordination for robotic grasping with deep learning and large-scale data collection,” The International Journal of Robotics Research, vol. 37, no. 4—5, pp. 421–436, 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 Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 23–30, 2017.
[3] N. Rudin, D. Hoeller, P. Reist and M. Hutter, “Learning to Walk in minutes using massively parallel deep reinforcement learning,” arXiv:2109.11978, 2021.
[4] B. Fang, S. Jia, Di Guo, Muhua Xu, Shuhuan Wen and F. Sun, “Survey of imitation learning for robotic manipulation,” 2019 International Journal of Intelligent Robotics and Applications, 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 Computing Surveys, vol. 50, no. 2, pp. 35, 2017.
[6] Z. Zhu, H. Hu “Robot Learning from Demonstration in Robotic Assembly: A Survey,” Robotics, Vol. 7, No. 2, 2018.
[7] P. Kormushev, S. Calinon and D. G. Caldwell, “Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input,” Advanced Robotics, 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 Proceedings of 2015 IEEE International Conference on Robotics and Automation, pp. 177–184, 2015.
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 III.

| Prop. 1 | Prop. 2 |
|--------|--------|
| F2FL   | FL2FL  |
| Node:120 | Node:170 |
| LSTM Cells:1 | LSTM Cells:1 |

| Lower |
|-------|
| Node:100 | Node:160 |
| LSTM Cells:1 | LSTM Cells:1 |
| Node:170 | Node:170 |
| LSTM Cells:1 | LSTM Cells:1 |

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 III.

| CW-RNN | FS-RNN | MTRNN | LSTM |
|--------|--------|-------|------|
| F2FL | F2FL | F2FL | F2FL |
| module:4 | module:3 | Node:850 | Node:1150 |
| Node:360 | Node:270 | Node:150 | Node:550 |
| Node:50 | Node:60 | Node:150 | Node:50 |
| Node:slow:50 | Node:slow:50 | τ:fast:10 | τ:slow:30 |
| τ:fast:20 | τ:slow:20 | τ:slow:200 | τ:slow:100 |

| Upper |
|-------|
| 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 a tilted plate [deg] | Prop. 1 | Prop. 2 | CW-RNN | FS-RNN | MTRNN | LSTM |
|------------------------------|--------|--------|--------|--------|-------|------|
| 20 | 100% | - | 100% | 0% | 40% | - | - | 100% | - | - |
| 15 | 100% | - | 100% | 0% | 60% | - | - | 80% | - | - |
| 10 | 80% | - | 80% | 20% | 0% | - | - | 100% | - | - |
| 5 | 100% | - | 60% | 80% | 80% | - | - | 100% | - | - |
| 0 | Success | Failure | Success Partial Success | Partial Success | Failure Failure Failure | Success Failure Failure Failure |

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

| Upper | Layer | Epoch | Time [s] |
|-------|-------|-------|----------|
| Node:360 | Node:270 | 39000 | 75324 |
| Node:360 | Node:450 | 9000 | 78321 |
| Node:360 | Node:550 | 15000 | 78321 |
| Node:360 | Node:650 | 21000 | 78321 |

[9] M. Edmonds, F. Gao, X. Xie, H. Liu, S. Qi, Y. Zhu, B. Rothrock and S. Zhang, “Feeling the force: Integrating force and pose for fluent discovery through imitation learning to open medicine bottles,” in Proceedings of 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3530–3537, 2017.

[10] L. Rozo, P. Jimenez and C. Torras, “A robot learning from demonstration framework to perform force-based manipulation tasks,” Intel Serv Robotics Vol. 6, No. 1, pp. 33–51, 2013.

[11] Khanh Nguyen, “Imitation Learning with Recurrent Neural Networks,” arXiv:1607.05241, 2016.

[12] H. Ravichandar, A.S. Polydoros, S. Chernova and A. Billard, “Recent Advances in Robot Learning from Demonstration,” Annual Review of Control, Robotics, and Autonomous Systems, Vol. 3, 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 Proceedings of the Conference on Robot Learning, pp. 1025–1037, 2020.

[15] S. E. Hihi and Y. Bengio, “Hierarchical recurrent neural networks for long-term dependencies,” Advances in Neural Information Processing Systems, pp. 493—499, 1995.

[16] J. Chung, S. Ahn, Y. Bengio, “Hierarchical Multiscale Recurrent Neural Networks,” in Proceedings of 5th International Conference on Learning Representations, 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 Proceedings of 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 10152–10157, 2020.

[18] T. Yu, P. Abbeel, S. Levine and C. Finn, “One-Shot Hierarchical Imitation Learning of Compound Visuomotor Tasks,” arXiv:1810.11043, 2018.

[19] S. Lee, S. Seo “Learning Compound Tasks without Task-specific Knowledge via Imitation and Self-supervised Learning,” in Proceedings of the 37th International Conference on Machine Learning, Vol. 119, pp. 5747–5756, 2020.

[20] Y. Wu and Y. Demiris, “Hierarchical Learning Approach for One-shot Action Imitation in Humanoid Robots,” in Proceedings of 11th International Conference on Control Automation Robotics & Vision, pp. 453–458, 2010.

[21] R. Fox, R. Berenstein, I. Stoica, K. Goldberg, “Multi-Task Hierarchical Imitation Learning for Home Automation,” in Proceedings of 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), pp. 1–8, 2019.

[22] S. Niekum, “Complex Task Learning from Unstructured Demonstrations,”
