Imitation Learning Based on Disentangled Representation Learning of Behavioral Characteristics

Ryoga Oishi[1], **Sho Sakaino**[2], **Toshiaki Tsuji**[1]
[1]Saitama University [2]University of Tsukuba
r.oishi@ms.saitama-u.ac.jp

Abstract: In the field of robot learning, coordinating robot actions through language instructions is becoming increasingly feasible. However, adapting actions to human instructions remains challenging, as such instructions are often qualitative and require exploring behaviors that satisfy varying conditions. This paper proposes a motion generation model that adapts robot actions in response to modifier directives—human instructions imposing behavioral conditions—during task execution. The proposed method learns a mapping from modifier directives to actions by segmenting demonstrations into short sequences, assigning weakly supervised labels corresponding to specific modifier types. We evaluated our method in wiping and pick-and-place tasks. Results show that it can adjust motions online in response to modifier directives, unlike conventional batch-based methods that cannot adapt during execution.

Keywords: Imitation learning, Disentangled representation learning

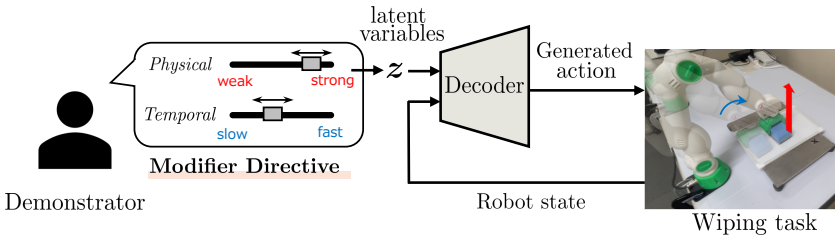


Figure 1: Overview of the proposed method. It generates the next motion trajectory based on a human-given **modifier directive** and the current robot state. In the example of wiping a whiteboard, a human gives **modifier directive** regarding the strength and speed of wiping, and the robot generates the corresponding motion.

1 Introduction

Imitation Learning (IL) [1] is a method that learns robot behavior based on human demonstration data and is positioned as one of the major approaches in robot learning, alongside reinforcement learning. IL is simple because it can learn a motion generation policy from demonstration data. It has been applied to various robot tasks such as folding tasks [2], assembly tasks [3], bottle-opening tasks [4], and handwriting tasks [5]. However, learning an environment-adapted motion generation policy from limited demonstrations is challenging, and some studies have pointed out the limitations of learning motions solely from demonstrations [6]. Therefore, many methods based on human intervention have been proposed as solutions to this issue [7, 8], with research advancing toward developing robots capable of executing complex tasks based on human instructions, linking human instructions to actions [9, 10].

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

Many studies that aim to associate human instructions with robot motion generation focus on the combination of large language models and existing motion generation models [11, 12, 13, 14, 15]. While these foundation models are highly effective for high-level task selection and coarse-grained skill switching, they still face significant challenges in performing online, fine-grained modulation of continuous motion parameters, such as speed, force, and smoothness. This limitation has also been highlighted in recent studies auditing robotic foundation models [16]. On the other hand, human instructions used for motion generation are often qualitative, making it difficult to establish a one-to-one mapping between the intended instruction and the resulting motion. For example, in the task of wiping a desk, instructions like "wipe strongly" or "wipe weakly" serve as modifiers that qualitatively adjust the force applied during motion execution. In this paper, such instructions will be referred to as **modifier directives**. It is particularly challenging not only to learn the basic capability of generating a wiping motion, but also to adjust the force applied during the motion according to such directives [17].

One method to directly link modifiers and learned representations is Disentanglement Representation Learning (DRL) [18]. DRL aims to separate the factors contained in the learning data and encode them as independent, distinct latent variables in the representation space [19, 20]. Within DRL, methods have been proposed that impose constraints on the learning of the latent space to separate the desired representations [21, 22], and approaches for applying these methods to robot learning have also been proposed [23]. An application to imitation learning involves the use of weakly supervised labels to impose constraints on the latent space representation [24, 25]. By assigning modifier directives as weakly supervised labels, it becomes possible to learn the correspondence between modifier directives and latent representations. By controlling these latent variables, robot motion trajectories corresponding to the modifier directives can be generated as outputs of the policy.

Many methods based on DRL for learning the correspondence between robot motions and modifier directives involve architectures that learn motions in a batch, and the application to online settings has not been explored. In real-world tasks, it is often expected that motion needs to be adjusted based on modifier directives during task execution, making it crucial to have a motion generation model capable of responding to such directives online. We define online motion generation as a process that dynamically adjusts a robot's motion in response to human-issued modifier directives during task execution. Achieving this requires a learning architecture that can learn the latent space corresponding to modifier directives and generate motions online based on that latent space.

This study proposes a motion generation model that allows robots to change their behavior online based on modifier directives. The study addresses the following two challenges: **1. Acquiring latent representations that link modifier directives and motions.** Generally, human motions targeted for imitation are high-dimensional, making direct control difficult. Therefore, it is desirable to manipulate motions through the latent variables of an autoencoder. However, these latent variables tend to lack interpretability, making it difficult to interpret and control them directly. To address this, we use DRL with weak supervision to learn a latent space corresponding to modifier directives, enabling motion editing based on those directives. **2. Adapting to modifier directives during online motion generation.** In order to achieve the desired motion when modifier directives are given during task execution, it is necessary to map the modifier directives to robot actions and generate the next action. During online execution, the system generates smooth trajectories by applying a weighted average to the predicted action sequences.

In the proposed method, motions are divided into short action sequences, and weakly supervised labels reflecting the modifier directives are assigned to each sequence. During model training, the error between the latent variables and the weakly supervised labels, as well as the output, is back-propagated to constrain the learning of the latent representation, establishing the correspondence between the generated sequences and the modifier directives. Through this approach, the method simultaneously addresses the two challenges: linking modifier directives to latent representations and enabling online motion generation based on modifier directives.

The main contributions of this paper are as follows:

- We propose a motion generation model that can modify motions online based on modifier directives, and demonstrate the applicability of latent representation learning for associating these directives with motions, compared to existing imitation learning approaches.
- By using a structure similar to Action Chunking for the predicted action sequences, we show that the desired motion can be achieved when modifier directives are input during online motion generation.

2 Related Work

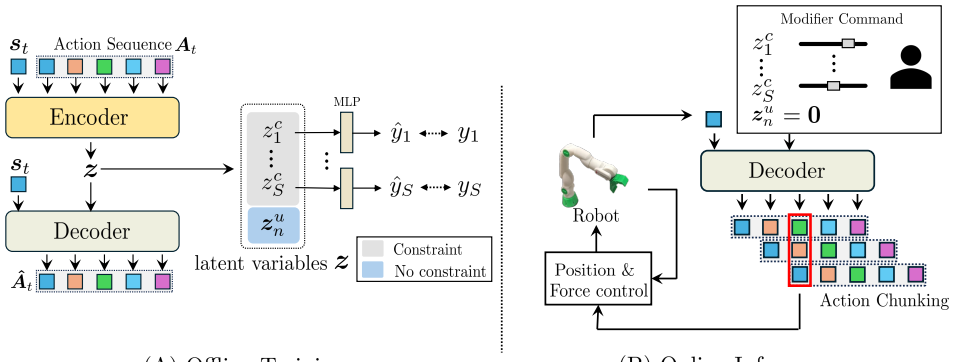
Imitation learning in robot manipulation. Imitation learning (IL) [1] is a fundamental approach in robot learning that trains robots from human demonstrations. A representative example is Behavior Cloning (BC), and due to the simplicity of its learning process, a variety of architectures have been proposed. Traditionally, learning methods using recurrent neural networks such as RNN [26] and Long Short Term Memory (LSTM) [27], which can take temporal information into account, have been extensively studied. More recently, Transformer [28] based models, such as Action Chunking with Transformers (ACT) [29], have also been proposed. ACT generates a chunk of action sequences and achieves smooth motion by calculating a weighted average with past action histories. By extending ACT to incorporate language instructions as model inputs, methods enabling motion generation based on human instructions have also been proposed [30]. However, these methods mainly focus on selecting appropriate skills or tasks based on language instructions and do not explicitly handle motion generation that reflects adjustments specified by modifier directives.

Motion Latent Space Manipulation. To efficiently learn the variations in human demonstration motions, it is effective to compress the demonstrations into low-dimensional information and learn their latent representations as distributions. Representative examples include studies embedding task skills into a latent space [31, 32]. Learning models often employ methods such as Variational AutoEncoder (VAE) [33] and Conditional Variational AutoEncoder (CVAE) [34], which assume distributions in the latent space. Several approaches have been proposed that constrain the learning of the latent space to acquire desired representations. These include representations related to physical constraints [35, 36], representations related to skills and motion primitives [32, 37, 38, 39], and representations related to motion styles [24, 40, 25]. Among these, the studies [24, 40, 25] are most closely related to this work, as they explore methods for learning interpretable or controllable latent spaces from demonstrations using weak supervision. However, these models primarily support feedforward motion generation, where motion is generated in a one-shot manner based on selected latent variables or styles, and they do not enable online modification or continuous editing of motion during execution.

3 Proposed Method

The purpose of this research is to develop an online motion generation model that can respond to modifier directives. The proposed model takes as input the robot state $\boldsymbol{s}_t$ and a command value $\boldsymbol{z}$ representing the latent variable corresponding to the modifier directive, and generates motions dynamically according to these inputs. For example, in a wiping task, the robot state reflects information obtained through physical interaction with the board, while the modifier directive specifies behavioral characteristics such as wiping strength (e.g., "wipe strongly" or "wipe weakly") or motion speed (e.g., "wipe slowly" or "wipe fast").

Fig. 2 shows the overview of the proposed learning model. Fig. 2(A) shows the offline learning architecture, while Fig. 2(B) represents the overview of online inference. As shown in Fig. 2(A), the proposed learning model primarily consists of two learning architectures: (1) an architecture that learns through CVAE with action sequences $\boldsymbol{A}_t$ as input, and (2) an architecture for associating action sequences $\boldsymbol{A}_t$ with modifier directives. In the first architecture, a CVAE is used to build a motion generation model that learns the distribution of motion sequences starting from a specific initial state $\boldsymbol{s}_t$. In the second architecture, constraints are imposed on the CVAE's latent variables



(A) Offline Training (B) Online Inference

Figure 2: (A) Overview of the offline learning architecture. (B) Overview of online inference.

z to disentangle them according to the features of action sequences A_t. This enables each disentangled latent variable to be trained to correspond to modifier directives. By editing variables in this learned latent space, motion generation that can control the robot actions according to given modifier directives becomes possible.

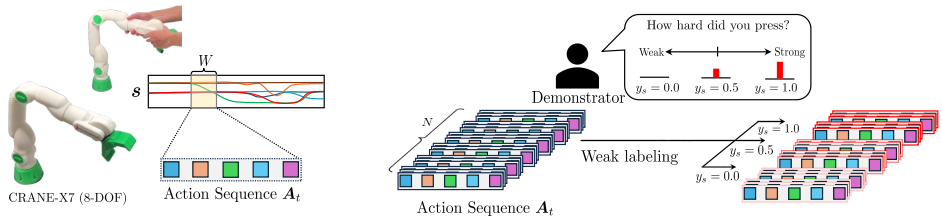
3.1 Preliminaries

Data Collection Method To collect human motion data, we use a teleoperation method based on bilateral control. Bilateral control involves two robots—a leader and a follower—where a human operates the leader robot, and the follower robot interacts with the environment. This setup enables the collection of motion data that accounts for the dynamics of the environment [41]. In particular, four-channel bilateral control [42], which simultaneously performs both position and force control on the leader and the follower, is effective for collecting data involving contact with the environment because it allows force information to be captured simultaneously [43, 44]. Its application to imitation learning has also been widely reported [44, 45, 46].

Robot System and Data Format For data collection and motion reproduction, we use the CRANE-X7 robot manufactured by RT Corporation. The CRANE-X7 has eight degrees of freedom, including the hand. During motion recording, each joint's angle q_t [rad], angular velocity $\dot{q}_t$ [rad/s], and torque τ_t [N·m] at time step t are recorded as the state $s_t = [q_t, \dot{q}_t, \tau_t]$. A sequence of demonstrations is defined as a time series of robot states, $\xi = \{s_1, s_2, \ldots, s_T\}$, where T is the total number of time steps in the demonstration. The collected motion data are organized into a dataset, $\mathcal{D} = \{\xi_i\}_{i=1}^{M}$, where M is the number of demonstrations.

3.2 Policy Learning

In this section, we describe the learning architecture of the motion generation model. First, the collected motion data is divided into multiple action sequences $A_t = [s_t, s_{t+1}, \ldots, s_{t+W-1}]$ using a sliding window of width W. The input to the learning model is the action sequence A_t, and the output is the predicted action sequence $\hat{A}_t$. In the CVAE, the encoder maps the action sequence A_t to the parameters of a probability distribution in the latent space. The latent variable z is then sampled from this distribution, and the decoder learns to reconstruct A_t from this sampled z. The reconstruction error $\mathcal{L}_{rec}$ is defined as the mean squared error of the predicted state vectors and actual state vectors over the entire sequence of width W, as shown in equation (1), where $\hat{s}_{t+j}$ denotes the predicted robot state at time $t+j$ generated by the decoder of the CVAE. The conditional variable for CVAE is the first sample of the action sequence s_t, and the model learns the distribution of action sequences A_t starting from the initial state s_t. The latent variable z is assumed to follow a diagonal Gaussian prior distribution $p(z) = \mathcal{N}(0, \mathbf{I})$. Thus, the KL regularization loss $\mathcal{L}_{kl}$, defined as the KL divergence from the approximate posterior $q_\phi(z|A_t, s_t)$ to the prior $p(z)$, is shown in equation (2).



(A) Data collection (Bilateral Control) (B) Weak supervision labeling of modifier directive

Figure 3: (A) Data collection through bilateral control. (B) Weakly supervised labeling of modifier directive by the demonstrator. In this example, labeling is performed for pressing force as follows: strong: $y_s = 1.0$, moderate: $y_s = 0.5$, weak: $y_s = 0.0$.

$$\mathcal{L}_{rec} = \frac{1}{W} \sum_{j=0}^{W-1} (\boldsymbol{s}_{t+j} - \hat{\boldsymbol{s}}_{t+j})^2 \tag{1}$$

$$\mathcal{L}_{kl} = D_{kl} \left[q_\phi(\boldsymbol{z} | \boldsymbol{A}_t, \boldsymbol{s}_t) \parallel \mathcal{N}(0, \mathbf{I}) \right] \tag{2}$$

3.3 Disentangled Representation Learning Using Weakly Supervised Labels

In this section, we describe the method for associating the latent representations of CVAE with modifier directives. We assume S types of modifier directives associated with the motion data and assign weakly supervised labels $y_s \in \{0.0, 0.5, 1.0\}$ for each type $(s = 1, 2, \ldots, S)$. The values are ordered by directive intensity. For example, in the case of a temporal directive representing motion speed, normal-speed motion is labeled with $y_s = 0.5$, fast motion with $y_s = 1.0$, and slow motion with $y_s = 0.0$. These modifier directives, such as "fast" and "slow," are annotated by the demonstrator, as shown in Fig. 3.

To disentangle the behavioral characteristics associated with modifier directives embedded in human motion data, we impose constraints on a subset of the latent variables using weakly supervised labels. We divide the latent variables into two groups: the constrained part $\boldsymbol{z}_s^c$ for modifier directives, and the unconstrained part $\boldsymbol{z}_n^u$, as defined in equation (3). Here, S denotes the number of modifier directives, and N represents the number of unconstrained latent variables. Each constrained latent variable $\boldsymbol{z}_s^c$ is fed into a multi-layer perceptron (MLP) with ReLU activation, whose output is denoted by $\hat{y}_s$, and is trained to predict the corresponding weakly supervised label y_s. This process imposes supervision on $\boldsymbol{z}_s^c$ such that each dimension of the constrained latent variables encodes the demonstrator's intended modifier directives during training. The training objective for this classification task is the binary cross-entropy loss $\mathcal{L}_{bce}$, as defined in equation (4), where $\sigma(\cdot)$ denotes the sigmoid function. For each modifier, the total loss is computed by summing the individual losses across all associated latent variables, as defined in equation (5).

$$\boldsymbol{z} = \{\boldsymbol{z}_s^c, \boldsymbol{z}_n^u\} = \{\underbrace{z_1^c, \ldots, z_S^c}_{\text{constraint}}, z_1^u, \ldots, z_N^u\} \tag{3}$$

$$\mathcal{L}_s = \mathcal{L}_{bce}(y_s, \hat{y}_s) = -[y_s \cdot \log(\sigma(\hat{y}_s)) + (1 - y_s) \cdot \log(1 - \sigma(\hat{y}_s))] \tag{4}$$

$$\mathcal{L}_{modi} = \sum_{s=1}^{S} \mathcal{L}_s \tag{5}$$

$$\mathcal{L} = \alpha \mathcal{L}_{rec} + \beta \mathcal{L}_{kl} + \gamma \mathcal{L}_{modi} \tag{6}$$

The overall training loss $\mathcal{L}$ is computed as a weighted sum of the reconstruction loss $\mathcal{L}_{rec}$, the KL regularization loss $\mathcal{L}_{kl}$, and the modifier classification loss $\mathcal{L}_{modi}$, as shown in equation (6). The coefficients α, β, and γ are weighting parameters that control the contribution of each loss term. In other words, training is guided by a trade-off between learning the distribution of action sequences $\boldsymbol{A}_t$ from a given initial state $\boldsymbol{s}_t$ and predicting the appropriate modifier directives. Minimizing this objective encourages a latent representation that is disentangled with respect to the weakly supervised labels provided by the demonstrator.

3.4 Online Inference

As shown in Fig. 2(B), during online motion generation, the robot state $\boldsymbol{s}_t$ and the modifier directives specified by a demonstrator are provided as inputs. The modifier directives are mapped to the latent representation $\boldsymbol{z} = \boldsymbol{z}_{\text{cmd}} = \{\boldsymbol{z}_s^c, \boldsymbol{z}_n^u\}$, which corresponds to the structure learned during training. The decoder then generates the action sequence $\hat{\boldsymbol{A}}_t$ with a length of $W(= 50)$ based on these inputs. Among the latent variables, the unconstrained component $\boldsymbol{z}_n^u$ is fixed at $\mathbf{0}$, while the constrained component $\boldsymbol{z}_s^c$ is manually adjusted to reflect the desired modifier directives. As shown in equation (7), the next robot state $\hat{\boldsymbol{s}}_{t+1}$ is computed as a weighted average of the generated action sequence $\hat{\boldsymbol{A}}_t$ and previous generations. Here $\hat{\boldsymbol{A}}_i[k]$ denotes the predicted robot state at relative step k in the predicted sequence starting from time i. For $t = 0$, our implementation applies a special-case handling without averaging and directly sets $\hat{\boldsymbol{s}}_1 = \hat{\boldsymbol{A}}_0[1]$.

$$\hat{\boldsymbol{s}}_{t+1} = \frac{\sum_{i=1}^{\min(t,\, W-1)} w_i \, \hat{\boldsymbol{A}}_{t+1-i}[i]}{\sum_{i=1}^{\min(t,\, W-1)} w_i}, \qquad w_i = \frac{1}{\log(i+1)}. \tag{7}$$

The role of the weighted average with past outputs is not only to address the non-Markovian nature of motion, as highlighted in prior work [29], but also to mitigate abrupt transitions that may occur when $\boldsymbol{z}_{\text{cmd}}$ is modified during online motion generation. Furthermore, the robot is controlled via position and force control using the computed $\hat{\boldsymbol{s}}_{t+1}$ as the command signal. The details of the command signal update frequency and robot control are described in the APPENDIX B.1. In this way, by providing command values $\boldsymbol{z}_{\text{cmd}}$ to the latent variables corresponding to modifier directives, the robot can autonomously generate motions that reflect the given directives.

4 Experiment

In the experiments, we provided $\boldsymbol{z}_{\text{cmd}}$ corresponding to the modifier directives as input and verified that the generated motions successfully reflected those modifier directives. Additionally, we demonstrate the effectiveness of Action Chunking in our method, where $\boldsymbol{z}_{\text{cmd}}$ is supplied during online motion generation. To validate these claims, we conducted two types of experiments. The first experiment assessed whether the generated motions were aligned with the given modifier directives in the task of wiping a whiteboard, utilizing two different types of modifier directives. The second experiment investigated the effectiveness of Action Chunking by comparing variations in the presence and type of weighted averaging applied in the proposed method.

4.1 Evaluation Metrics

We quantitatively assessed performance using two complementary metrics: **1) Task Success Rate (TSR):** this metric evaluates whether the generated motion successfully completes the intended task. For each task, we defined specific success criteria to determine completion. **2) Modifier Directive Errors (MDEs):** this metric quantifies the alignment between the modifier directives and the resulting motion. It measures the discrepancy between intended modifications and the generated motion across each latent variable dimension. Detailed formulations for both metrics are provided in APPENDIX B.2.

4.2 Comparison Methods

To evaluate our proposed approach, we implemented two distinct imitation learning architectures. The first is **ACT** [29], a CVAE model using Transformer architecture for both encoder and decoder. The second comparison method is **CVAE-LSTM**, a CVAE model employing LSTM networks for both. Both methods incorporate Action Chunking on the generated sequence. For our proposed extensions, we developed **ACT (Proposed)** and **CVAE-LSTM (Proposed)**, which extend the original models with a latent representation learning constraint. This constraint predicts weakly supervised labels from the latent variable $\boldsymbol{z}$ and backpropagates the error. By comparing each architecture with and without our proposed extension, we assess both the effectiveness of our approach and its adaptability to different models when responding to modifier directives during online execution.

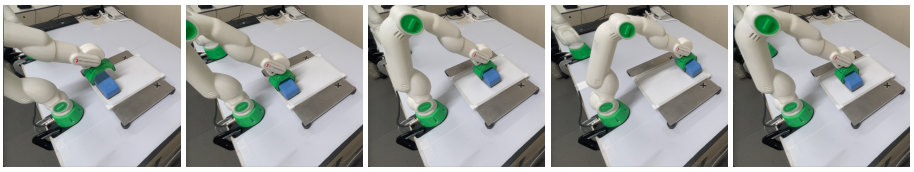


Figure 4: Wiping task: the robot grabbed the whiteboard eraser and uses its entire body and joints to wipe the whiteboard.

4.3 Experimental Setup

We evaluated our approach on the Wiping task, focusing on temporal and physical modifier directives. As illustrated in Fig. 4, this task involves grasping a whiteboard eraser and wiping a whiteboard positioned at a fixed location. Temporal directives were defined based on the time required to complete three wiping cycles and evaluated under three conditions: Slow, Moderate, and Fast. Physical directives were characterized by the wiping pressure applied to the whiteboard, also evaluated under three conditions: Weak, Moderate, and Strong. Table 1 shows the correspondence between these modifier directives and their associated weakly supervised labels. The latent space dimension was set to 3, comprising $S = 2$ latent variables constrained by weak supervision labels and $N = 1$ unconstrained latent variable. In the Wiping task, z_1^c corresponds to physical directives and z_2^c to temporal directives, both constrained by labels during training.

Table 1: Correspondence table between modifier directive and weakly supervised labels

	$y_s = 0.0$	$y_s = 0.5$	$y_s = 1.0$
Physical (phys)	Weak	Moderate	Strong
Temporal (temp)	Slow	Moderate	Fast

5 Results

5.1 Motion Generation Using Modifier Directives

To evaluate whether the generated motions accurately reflected the intended modifier directives, we computed the modifier directive errors (MDEs) for each latent variable based on the consistency between command variations and the intended directives. Table 2 presents the Task Success Rate (TSR) and MDEs for the Wiping task. For the MDEs, lower values indicate that the latent variable more clearly represents the intended modifier. Latent variables with a compatibility score below 0.50 are highlighted in bold.

For the proposed method, we set the latent command to $\boldsymbol{z}_{\text{cmd}} = \{z_1, z_2, z_3\} = \{z_1^c, z_2^c, z_1^u\}$ and tested axis-aligned patterns in which exactly one constrained coordinate (z_1^c, z_2^c) was set to $\pm\{1.0, 2.0\}$ while the others were fixed at 0.0; the unconstrained variable z_1^u was always 0.0 (9 settings, including a neutral setting of all zeros). This approach focuses the evaluation on the two dimensions explicitly constrained to represent modifier semantics. In contrast, for the baseline methods (CVAE-LSTM and ACT), we used $\boldsymbol{z}_{\text{cmd}} = \{z_1^u, z_2^u, z_3^u\}$ and varied exactly one coordinate to $\pm\{1.0, 2.0\}$ while fixing the other two at 0.0 (13 settings, including a neutral setting of all zeros). Since the baselines lack the proposed disentanglement constraint, it is not known which latent dimension corresponds to which modifier. Therefore, we evaluate all three dimensions.

The proposed method achieved higher task success rates compared to the baseline models, demonstrating its ability to generate motions aligned with the provided directives. Specifically, the standard CVAE-LSTM exhibited MDEs exceeding 1.00 across all latent variables, with minimal differentiation between z_1, z_2, and z_3. This suggests limited capability in associating individual latent dimensions with distinct directive types, indicating an entangled representation. In contrast, CVAE-LSTM (Proposed) achieved substantially lower MDEs for the latent variables explicitly constrained

Table 2: Success rate and modifier directive conformity Index in the Wiping task

	Task success rate (TSR)	z_1(phys)		z_2(temp)		z_3	
		temp	phys	temp	phys	temp	phys
CVAE-LSTM	100.0% [13/13]	1.09	1.01	1.06	1.04	1.11	0.92
CVAE-LSTM (Proposed)	100.0% [9/9]	0.69	0.63	**0.22**	0.63	–	–
ACT	0.0% [0/13]	–	–	–	–	–	–
ACT (Proposed)	88.9% [8/9]	**0.43**	1.47	1.00	1.95	–	–

by modifier directives, with the temporal MDE for z_2 dropping to 0.22 and the physical MDE for z_1 to 0.63. These reductions suggest that each latent variable more clearly and independently encodes a specific directive, supporting improved disentanglement through the proposed approach. Nevertheless, the temporal MDE for z_1 remained at 0.69, implying some residual entanglement between latent variables. Similarly, ACT (Proposed) showed improved disentanglement compared to the original ACT, with a temporal MDE of 0.43 when varying z_1. However, the physical MDE for z_2 remained high (1.95). This is contrary to the intended behavior where z_1 should represent temporal directives and z_2 should represent physical directives. This suggests that modifier effects may still appear along unintended latent axes. This result highlights that, while the proposed modifications enhanced directive alignment, there remains room for improving the orthogonality and independence of latent variable representations. These findings confirm that the proposed method can effectively generate motions that respond to temporal and physical modifiers in the Wiping task, demonstrating a clear correspondence between latent variables and modifier semantics.

5.2 Effect of Action Chunking on Motion Stability

To demonstrate the effectiveness of Action Chunking in our method, where $\boldsymbol{z}_{\text{cmd}}$ is supplied during online motion generation, we evaluated its impact on motion stability in the Wiping task. Table 3 summarizes the task success rate (TSR) under different temporal weighting functions in action chunking. We compared the case without action chunking (No weight) against three weighting schemes. When action chunking was not applied, the predicted actions exhibited severe oscillations during execution, resulting in a 0% TSR. In contrast, applying action chunking substantially improved motion stability and task performance. Notably, the use of $w_i = 1/\log(i+1)$ achieved a 100.0% success rate for both the CVAE-LSTM and ACT variants of our method. This result highlights the importance of temporal smoothing in mitigating abrupt transitions in the predicted action sequence caused by changes in the latent command $\boldsymbol{z}_{\text{cmd}}$. These findings empirically support the effectiveness of our proposed Action Chunking mechanism in enhancing online motion generation under modifier directives.

Table 3: Relationship between weight parameters and task success rate (TSR) in action chunking

	No weight	$w_i = 1/(i+1)$	$w_i = \exp(-m * i)$	$w_i = 1/\log(i+1)$
CVAE-LSTM (Proposed)	0.0% [0/5]	**100.0%** [5/5]	0.0% [0/5]	**100.0%** [5/5]
ACT (Proposed)	0.0% [0/5]	0.0% [0/5]	**100.0%** [5/5]	**100.0%** [5/5]

6 Conclusion

In this study, we present a motion generation model capable of dynamically adjusting robot behaviors in response to online modifier directives. We propose a learning framework that links human-issued modifier directives with corresponding motion variations, enabling online adaptation of generated trajectories. Our evaluation demonstrates that the model effectively handles specific types of directives, such as temporal and physical constraints. Furthermore, by introducing an action chunking mechanism to mitigate abrupt transitions in motion sequences, we observe a notable improvement in task success rates. These results validate the proposed method's effectiveness in generating responsive and adaptable motions under modifier directives.

Acknowledgments

This paper is based on results obtained from a project, JPNP14004, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018. ISSN 1935-8253. doi:10.1561/2300000053. URL http://dx.doi.org/10.1561/2300000053.

[2] P.-C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata. Repeatable folding task by humanoid robot worker using deep learning. *IEEE Robotics and Automation Letters*, 2(2): 397–403, 2017. doi:10.1109/LRA.2016.2633383.

[3] Y. Wang, C. C. Beltran-Hernandez, W. Wan, and K. Harada. Robotic imitation of human assembly skills using hybrid trajectory and force learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11278–11284, 2021. doi:10.1109/ICRA48506.2021.9561619.

[4] K. Takeuchi, S. Sakaino, and T. Tsuji. Motion generation based on contact state estimation using two-stage clustering. *IEEJ Journal of Industry Applications*, 12(5):1000–1007, 2023. doi:10.1541/ieejjia.22012635.

[5] K. Kutsuzawa and M. Hayashibe. Imitation learning with time-varying synergy for compact representation of spatiotemporal structures. *IEEE Access*, 11:34150–34162, 2023. doi:10.1109/ACCESS.2023.3264213.

[6] E. Bıyık, D. P. Losey, M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67, 2022. doi:10.1177/02783649211041652. URL https://doi.org/10.1177/02783649211041652.

[7] S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011. URL https://arxiv.org/abs/1011.0686.

[8] A. Mandlekar, D. Xu, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese. Human-in-the-loop imitation learning using remote teleoperation, 2020. URL https://arxiv.org/abs/2012.06733.

[9] H. Zhou, Z. Bing, X. Yao, X. Su, C. Yang, K. Huang, and A. Knoll. Language-conditioned imitation learning with base skill priors under unstructured data. *IEEE Robotics and Automation Letters*, 9(11):9805–9812, 2024. doi:10.1109/LRA.2024.3466076.

[10] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation, 2021. URL https://arxiv.org/abs/2109.12098.

[11] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. B. Amor. Language-conditioned imitation learning for robot manipulation tasks, 2020. URL https://arxiv.org/abs/2010.12083.

[12] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning, 2022. URL https://arxiv.org/abs/2202.02005.

[13] T. Kobayashi, M. Kobayashi, T. Buamanee, and Y. Uranishi. Bi-lat: Bilateral control-based imitation learning via natural language and action chunking with transformers, 2025. URL https://arxiv.org/abs/2504.01301.

[14] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022. URL https://arxiv.org/abs/2204.01691.

[15] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL https://arxiv.org/abs/2307.15818.

[16] S. Karnik, Z.-W. Hong, N. Abhangi, Y.-C. Lin, T.-H. Wang, C. Dupuy, R. Gupta, and P. Agrawal. Embodied red teaming for auditing robotic foundation models, 2025. URL https://arxiv.org/abs/2411.18676.

[17] K. Kawaharazuka, Y. Kawamura, K. Okada, and M. Inaba. Imitation learning with additional constraints on motion style using parametric bias. *IEEE Robotics and Automation Letters*, 6 (3):5897–5904, 2021. doi:10.1109/LRA.2021.3087423.

[18] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives, 2014. URL https://arxiv.org/abs/1206.5538.

[19] F. Locatello, S. Bauer, M. Lucic, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. *CoRR*, abs/1811.12359, 2018. URL http://arxiv.org/abs/1811.12359.

[20] X. Wang, H. Chen, S. Tang, Z. Wu, and W. Zhu. Disentangled representation learning, 2024. URL https://arxiv.org/abs/2211.11695.

[21] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sy2fzU9gl.

[22] M. J. Vowels, N. C. Camgoz, and R. Bowden. Gated variational autoencoders: Incorporating weak supervision to encourage disentanglement, 2019. URL https://arxiv.org/abs/1911.06443.

[23] J. Hu, Z. Wang, P. Stone, and R. Martín-Martín. Disentangled unsupervised skill discovery for efficient hierarchical reinforcement learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ePOBcWfNFC.

[24] Y. Hristov and S. Ramamoorthy. Learning from demonstration with weakly supervised disentanglement. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ldau9eHU-q0.

[25] W. Song, S. Jeon, H. Choi, K. Sohn, and D. Min. Learning disentangled skills for hierarchical reinforcement learning through trajectory autoencoder with weak labels. *Expert Systems with Applications*, 230:120625, 2023. ISSN 0957-4174. doi:https://doi.org/10.1016/j.eswa.2023.120625. URL https://www.sciencedirect.com/science/article/pii/S0957417423011272.

[26] J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. ISSN 0364-0213. doi:https://doi.org/10.1016/0364-0213(90)90002-E. URL https://www.sciencedirect.com/science/article/pii/036402139090002E.

[27] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

[28] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. URL https://api.semanticscholar.org/CorpusID:13756489.

[29] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

[30] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking, 2023. URL https://arxiv.org/abs/2309.01918.

[31] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rk07ZXZRb.

[32] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play, 2019. URL https://arxiv.org/abs/1903.01973.

[33] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. URL https://api.semanticscholar.org/CorpusID:216078090.

[34] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *Neural Information Processing Systems*, 2015. URL https://api.semanticscholar.org/CorpusID:13936837.

[35] K. Kutsuzawa, S. Sakaino, and T. Tsuji. Sequence-to-sequence model for trajectory planning of nonprehensile manipulation including contact model. *IEEE Robotics and Automation Letters*, 3(4):3606–3613, 2018. doi:10.1109/LRA.2018.2854958.

[36] R. Okumura, N. Nishio, and T. Taniguchi. Tactile-sensitive newtonianvae for high-accuracy industrial connector insertion. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4625–4631, 2022. doi:10.1109/IROS47612.2022.9981610.

[37] M. Noseworthy, R. Paul, S. Roy, D. Park, and N. Roy. Task-conditioned variational autoencoders for learning movement primitives. In *Conference on Robot Learning*, 2019. URL https://api.semanticscholar.org/CorpusID:210178479.

[38] T. Osa and S. Ikemoto. Goal-conditioned variational autoencoder trajectory primitives with continuous and discrete latent codes. *SN Computer Science*, 1 (5), Sept. 2020. ISSN 2661-8907. doi:10.1007/s42979-020-00324-7. URL http://dx.doi.org/10.1007/s42979-020-00324-7.

[39] D. Tanneberg, K. Ploeger, E. Rueckert, and J. Peters. Skid raw: Skill discovery from raw trajectories. *IEEE Robotics and Automation Letters*, 6(3):4696–4703, 2021. doi:10.1109/LRA.2021.3068891.

[40] H. Nemlekar, R. R. Sanchez, and D. P. Losey. Pecan: Personalizing robot behaviors through a learned canonical space, 2024. URL https://arxiv.org/abs/2407.16081.

[41] Y. Saigusa, S. Sakaino, and T. Tsuji. Imitation learning for nonprehensile manipulation through self-supervised learning considering motion speed. *IEEE Access*, 10:68291–68306, 2022. doi:10.1109/ACCESS.2022.3185651.

[42] S. Sakaino, T. Sato, and K. Ohnishi. Multi-dof micro-macro bilateral controller using oblique coordinate control. *IEEE Transactions on Industrial Informatics*, 7(3):446–454, 2011. doi:10.1109/TII.2011.2158837.

[43] A. Sasagawa, K. Fujimoto, S. Sakaino, and T. Tsuji. Imitation learning based on bilateral control for human–robot cooperation. *IEEE Robotics and Automation Letters*, 5 (4):6169–6176, Oct. 2020. ISSN 2377-3774. doi:10.1109/lra.2020.3011353. URL http://dx.doi.org/10.1109/LRA.2020.3011353.

[44] M. Kobayashi, T. Buamanee, and T. Kobayashi. Alpha- α and bi-act are all you need: Importance of position and force information/ control for imitation learning of unimanual and bimanual robotic manipulation with low-cost system. *IEEE Access*, 13:29886–29899, 2025. doi:10.1109/ACCESS.2025.3541200.

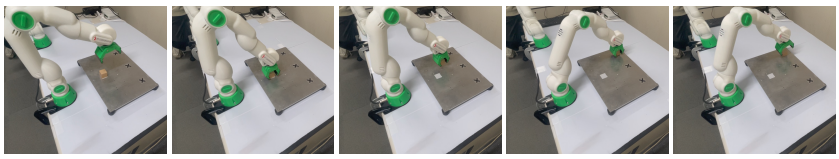
[45] K. Yamane, Y. Saigusa, S. Sakaino, and T. Tsuji. Soft and rigid object grasping with cross-structure hand using bilateral control-based imitation learning. *IEEE Robotics and Automation Letters*, 9(2):1198–1205, 2024. doi:10.1109/LRA.2023.3335768.

[46] T. Tsuji. Mamba as a motion encoder for robotic imitation learning. *IEEE Access*, 13:69941–69949, 2025. doi:10.1109/ACCESS.2025.3561283.

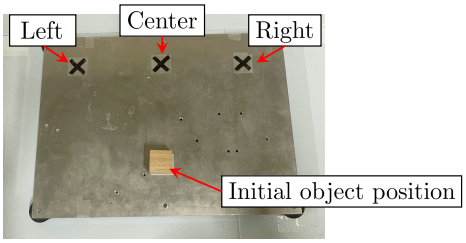
[47] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL https://api.semanticscholar.org/CorpusID:6628106.

A LIMITATION

Although the proposed method enables motion generation that can respond to online modifier directives, there remain several limitations. First, while our method demonstrates successful disentanglement between latent variables and modifier directives in certain tasks, this property does not generalize consistently across directive types or task domains. As an additional evaluation, we applied our framework to a Pick-and-Place task, where the robot is required to grasp an object and place it at a designated location, as shown in Fig. 5(A). This task involves two types of modifier directives: temporal (Slow, Moderate, Fast) and spatial (Left, Center, Right), with the latter specifying the object placement location, as shown in Fig. 5(B).



(A) Snapshots of online robot execution during the Pick-and-Place task.



(B) Top view of workspace for the Pick-and-Place task.

Figure 5: Pick-and-Place task: The robot picks up a block and places it at a designated location.

Table 4: Success rate and modifier directive conformity index in the Pick-and-Place task

	Task success rate (TSR)	z_1		z_2		z_3	
		temp	spat	temp	spat	temp	spat
CVAE-LSTM	100.0% [13/13]	1.05	**0.05**	1.02	0.67	1.01	1.38
CVAE-LSTM (Proposed)	100.0% [9/9]	1.09	1.04	1.35	2.06	–	–
ACT	100.0% [13/13]	1.00	1.04	1.05	1.00	0.98	1.04
ACT (Proposed)	100.0% [9/9]	1.24	0.97	1.00	1.00	–	–

As shown in Table 4, all models achieved high task success rates; however, the MDEs exceeded 1.00 for most directive-latent variable combinations. A notable exception was observed in the CVAE-LSTM model, where varying z_1 yielded a very low spatial MDE of 0.05. Nevertheless, this effect was not consistently reproduced across other models or variables. These findings suggest that spatial directives—being symbolic and discrete in nature—are not well captured by the current framework, which is based on continuous latent variables and weak supervision. This limitation highlights the need to develop more flexible mechanisms to accommodate the semantic nature of different directives. In particular, bridging the mismatch between discrete directive labels and continuous latent representations may require the introduction of additional architectural elements, such as gating mechanisms [22] that explicitly handle symbolic task features.

Second, the weighting parameters α, β, and γ used in the loss function significantly affect training outcomes. In this study, we empirically tuned these weights individually for each task. However, the importance of each component—reconstruction, KL divergence, and modifier directive—may vary with the task type and directive semantics. Future work should explore task-adaptive or data-driven strategies to balance these components effectively.

Finally, our experiments were limited to proprioceptive input modalities: joint angles $\boldsymbol{q}$ [rad], angular velocities $\dot{\boldsymbol{q}}$ [rad/s], and joint torques $\boldsymbol{\tau}$ [N·m]. As a result, the evaluation was constrained to tasks where these modalities are the primary source of directive-related variation. In tasks such as polishing or inspection, directives like "carefully" or "thoroughly" may rely heavily on auditory or visual cues. Extending the framework to incorporate multimodal sensory inputs remains an important direction for future work.

B APPENDIX

B.1 Robot Control Configuration

Bilateral control for data collection was executed with a control period of 2 ms, and the data were collected at this frequency. The data were then downsampled every 20 steps and used for training. For autonomous operation, we followed the approach in where commands are given to the follower robot. The follower robot performed position and force control with a control period of 2 ms, and the command values inferred by the model were updated every 20 steps.

B.2 Evaluation Metrics Details

Task Success Rate (TSR). Task Success Rate (TSR) evaluates whether the robot successfully completes the assigned task, regardless of whether the generated motion reflects the intended modifier directive. For the Wiping task, a trial was considered successful if the robot could grasp the eraser from the initial pose and perform repeated back-and-forth wiping motions on the whiteboard. Failures were defined as the inability to grasp the eraser, premature stopping, or lifting the eraser off the whiteboard during execution. In the Pick-and-Place task, success was defined as grasping the object from the initial pose and placing it onto the designated workspace. Failure cases included missing the object during grasping, unintended gripper actuation mid-air, failure to open the gripper during placement, or unintentional stopping.

Modifier Directive Errors (MDEs). Modifier Directive Errors (MDEs) quantify the extent to which the generated motion aligns with a given modifier directive by comparing it to reference motions annotated with directive labels. For each directive type, we first collected reference motion data corresponding to three predefined levels (e.g., Slow, Moderate, Fast), as shown in Table 1. For each level, four trials were conducted, resulting in twelve reference samples. A task-relevant scalar feature was extracted from each trial and averaged across repetitions for each directive level. These three averaged feature values were then fitted to a line via least squares regression, yielding a reference line $y = ax + b$, where $x \in \{0.0, 0.5, 1.0\}$ represents the directive label normalized to the range $[0, 1]$, and y is the corresponding feature value.

The fitted coefficients a and b used for each directive in both tasks are shown in Tables 5 and 6. For instance, in the Pick-and-Place task, the slope and intercept for the temporal directive were -7.097 and 16.525, respectively, while for the spatial directive they were -0.725 and 3.526. In the Wiping task, the temporal directive yielded a slope of -6.379 and intercept of 12.414, and the physical directive produced values of -1.115 and -1.017.

In the experiment, the modifier directive was specified using five latent command values: $-2, -1, 0, 1, 2$. These values were linearly scaled to match the normalized directive space, resulting in $x \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$. The corresponding feature values y obtained from the generated motions were then fitted to a line $y = cx + d$ via least squares regression.

MDE is defined as the normalized Euclidean distance between the slopes and intercepts of the reference and generated lines, and is given by:

$$\text{MDE} = \sqrt{\left(\frac{a-c}{a}\right)^2 + \left(\frac{b-d}{b}\right)^2} \tag{8}$$

To avoid division by zero, we assume $a \neq 0$ and $b \neq 0$. Lower MDE values indicate better alignment between the intended directive and the generated motion. This metric is computed independently for each directive-latent variable pair to assess how well each latent dimension captures a specific type of directive.

The feature values y used for each directive type are defined as follows. For the Wiping task, the average time required to complete three wiping cycles was used for temporal directives, while for the Pick-and-Place task, the time from the start of the motion to placing the object at the designated position was used. For Physical directives, we recorded the minimum torque $\tau_4\,[\mathrm{Nm}]$ at joint 4 during each wiping cycle and computed the average over three cycles. For Spatial directives used in the Pick-and-Place task, we recorded the final joint angle $q_2\,[\mathrm{rad}]$ of joint 2 at the completion of the placement action.

Table 5: Reference line parameters (a, b) for the Pick-and-Place task

Modifier Type	Slope (a)	Intercept (b)
Temporal	-7.097	16.525
Spatial	-0.725	3.526

Table 6: Reference line parameters (a, b) for the Wiping task

Modifier Type	Slope (a)	Intercept (b)
Temporal	-6.379	12.414
Physical	-1.115	-1.017

B.3 Learning Method Details

Table 7 summarizes the training parameters and their values. Table 8, Table 9 and Table 10 show the hyperparameters used for the ACT model, the CVAE-LSTM model and the weak-label predictor, respectively. The weights of each loss component are presented in Table 11. The models were trained on a workstation equipped with an 11th Gen Intel (R) Core (TM) i7-11800H @ 2.30GHz CPU and an NVIDIA RTX A3000 Laptop GPU. The overall training objective follows the loss formulation defined in Equation (6), which combines the reconstruction loss $\mathcal{L}_{rec}$, the KL regularization loss $\mathcal{L}_{kl}$, and directive supervision loss $\mathcal{L}_{modi}$. The weights α, β, and γ control the contribution of each term and were determined empirically per task and model, as shown in Table 11. For baseline models (ACT and CVAE-LSTM), γ was not applied since modifier supervision was not used.

Table 7: Training parameters

Parameter	**Value**
Optimizer	Adam [47] ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
Learning rate	1.0×10^{-3}
Gradient clipping	1.0
Batch size	256
Epochs	1000 (Seq2Seq), 5000 (ACT)

Table 8: Hyperparameters of ACT

Hyperparameter	**Value**
Encoder layers	2
Decoder layers	3
Hidden dimension	48
Feedforward dimension	192
Number of heads	4
Window size($= W$)	50

Table 9: Hyperparameters of CVAE-LSTM

Hyperparameter	**Value**
Encoder layers	3
Decoder layers	3
Output layer (fully connected)	1
Hidden dimension	256
Window size($= W$)	50

Table 10: Hyperparameters of the weak-label predictor

Hyperparameter	**Value**
Input dimension	1
Hidden layers	2
Hidden widths	[3, 3]
Activation	ReLU
Output dimension	1

Table 11: Loss function weights (α, β, γ) used in each task and model

Task	**Model**	α	β	γ
Wiping	ACT	1.0	4.0	–
	ACT (Proposed)	1.0	1.0	0.5
	CVAE-LSTM	1.0	4.0	–
	CVAE-LSTM (Proposed)	1.0	0.3	2.5
Pick-and-Place	ACT	1.0	4.0	–
	ACT (Proposed)	1.0	2.0	5.0
	CVAE-LSTM	1.0	4.0	–
	CVAE-LSTM (Proposed)	1.0	0.3	2.5