CLASS: Contrastive Learning via Action Sequence Supervision for Robot Manipulation

Sung-Wook Lee, Xuhui Kang, Brandon Yang, Yen-Ling Kuo
University of Virginia
{dcs3zc,xuhui,branyang,ylkuo}@virginia.edu

Abstract: Recent advances in Behavior Cloning (BC) have led to strong performance in robotic manipulation, driven by expressive models, sequence modeling of actions, and large-scale demonstration data. However, BC faces significant challenges when applied to heterogeneous datasets, such as visual shift with different camera poses or object appearances, where performance degrades despite the benefits of learning at scale. This stems from BC's tendency to overfit individual demonstrations rather than capture shared structure, limiting generalization. To address this, we introduce Contrastive Learning via Action Sequence Supervision (CLASS), a method for learning behavioral representations from demonstrations using supervised contrastive learning. CLASS leverages weak supervision from similar action sequences identified via Dynamic Time Warping (DTW) and optimizes a soft InfoNCE loss with similarity-weighted positive pairs. We evaluate CLASS on 5 simulation benchmarks and 3 real-world tasks to achieve competitive results using retrieval-based control with representations only. Most notably, for downstream policy learning under significant visual shifts, Diffusion Policy with CLASS pre-training achieves an average success rate of 75%, while all other baseline methods fail to perform competitively.

Keywords: Robot Manipulation, Action Chunking, Supervised Contrastive Learning, Vision Representation Learning

1 Introduction

Behavior Cloning (BC) has demonstrated strong performance in robotic manipulation by leveraging expressive models and action sequence modeling. Efforts to improve BC have focused on large-scale dataset collection [1, 2] and advances in model architectures [3, 4, 5] to better capture the complex distribution of demonstration data. However, expressive policies often struggle to generalize, especially when trained on demonstrations collected under heterogeneous conditions—that is, where the policy must adapt to additional properties not present in homogeneous data, such as changes in viewpoint or object appearance [6, 7]. This suggests a tendency to overfit individual actions and a limited ability to capture shared structure across demonstrations [8].

To address this, we propose **Contrastive Learning via Action Sequence Supervision (CLASS)**, a framework for learning behaviorally grounded representations from demonstrations using supervised contrastive learning. Rather than relying on direct action prediction, CLASS supervises the encoder by aligning observations based on action sequence similarity, measured via Dynamic Time Warping (DTW), encouraging states that lead to similar future behaviors to cluster in the latent space. This weak supervision enables the model to capture shared structure across demonstrations, improving robustness to variations in visual conditions such as camera pose and object appearance. The learned representation supports both retrieval-based inference and policy fine-tuning, and consistently improves performance across both homogeneous and heterogeneous data settings. Across a range of simulated and real-world robotic manipulation tasks, CLASS achieves strong gains over behavior cloning and representation learning baselines, demonstrating its ability to learn more transferable and composable behavioral representations.

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

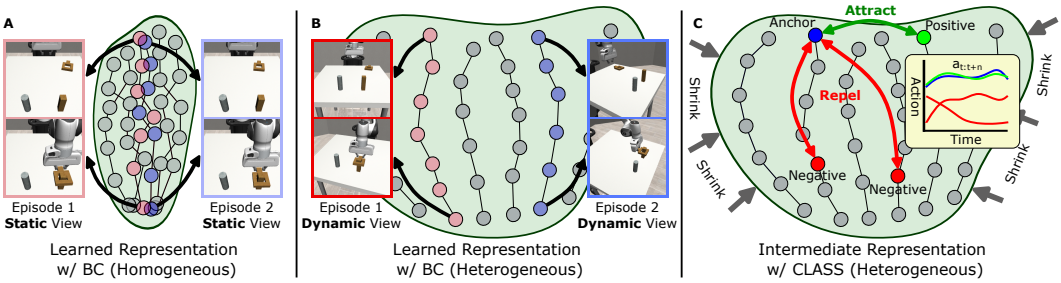


Figure 1: **Comparison between Behavior Cloning (BC) and Contrastive Learning via Action Sequence Supervision (CLASS). (A)** With homogeneous demonstrations with consistent visual conditions, BC learns a compact representation with high transferability. **(B)** With heterogeneous demonstrations such as varying viewpoints, BC overfits to individual state-action pairs and generalizes poorly. **(C)** CLASS addresses this by attracting states with similar action sequences and repelling those with dissimilar ones, with a soft supervised contrastive learning objective to learn more robust and composable representations.

Our contributions are as follows: 1) We introduce CLASS, a method for learning robust visual representations from demonstrations by aligning observations based on action sequence similarity. 2) We develop a novel scheme for soft contrastive learning that focuses on weighting similar action sequences according to their DTW-based similarity. 3) We show that CLASS outperforms behavior cloning and representation learning baselines across a range of simulated and real-world robotic manipulation tasks, particularly under heterogeneous visual conditions.

2 Related Work

Action Sequence Modeling (Action Chunking) Real-world physical tasks demand executing action sequences that are both temporally coherent and sensitive to system dynamics. However, traditional policy learning methods that rely on the Markov assumption and produce actions one step at a time have struggled with long-horizon tasks with compounding errors [9, 10]. Recent research addresses these issues by casting policy learning as a sequence modeling problem, an approach that has demonstrated strong empirical performance across domains [11, 12]. For behavior cloning in robot manipulation, Diffusion Policy (DP) [3] incorporates a diffusion framework with action chunking to achieve the state-of-the-art performance. Action Chunking Transformer [4] uses temporal ensembling between previously generated actions to further improve temporal consistency. In addition, VQ-BET [5] leverages Vector Quantization [13] to discretize continuous action chunks. These discrete representations are then integrated into the Behavior Transformer [14], leading to improvements in both inference speed and policy performance. While the previous action chunking methods rely on strong supervision via behavior cloning, our method focuses on using the similarity between action sequences as a contrastive signal for representation learning.

Contrastive Representation Learning Contrastive learning has been applied across different fields to extract informative representations from high-dimensional inputs. In reinforcement learning (RL), it is often used to encourage representations that capture task-relevant dynamics and improve sample efficiency. For example, [15] contrasts observations based on whether a sampled state is likely to occur in the future relative to a given anchor state. In imitation learning, time-contrastive objectives [16, 17, 18] within each demonstration are applied to learn visual representations for downstream robot manipulation tasks. In trajectory prediction, [19] proposes a hierarchical contrastive approach that uses high-level action information to bring similar behaviors closer in the embedding space. In autonomous driving, [20] uses pseudo-labeled actions from videos to identify positive pairs in a binary manner to perform contrastive learning on vision representations. More broadly, recent works [21, 22, 23, 24] explore soft contrastive learning by assigning continuous similarity-based weights, providing finer-grained learning signals that better capture inter-sample relationships. Unlike prior contrastive methods in robotics that focus on learning temporal relationships within individual demonstrations, our method composes similar behaviors across different demonstrations.

In addition, whereas most previous soft contrastive learning methods focus on assigning similarity-based weights to negative pairs, we focus on soft similarity weighting between positive pairs. This design choice aligns well with the goal of imitation learning and retrieval-based policy learning methods like VINN [25], where the quality of the learned representation depends on how well similar demonstrations are clustered.

3 Problem Statement

We address imitation learning by reformulating it as a behaviorally grounded representation learning problem. The goal is to learn an embedding space where observations with similar future behaviors are mapped close together, while dissimilar ones are separated. Instead of relying on direct action supervision, we propose to learn the embedding space via a soft supervised contrastive objective guided by trajectory-level action similarity.

Specifically, given a dataset of demonstration trajectories consisting of observation and action sequences, we define $\mathbf{O}_t = \{o_{t-T_o-1}, \ldots, o_{t-1}, o_t\}$ and $\mathbf{A}_t = \{a_t, a_{t+1}, \ldots, a_{t+T_p-1}\}$, where T_o is the number of past observations and T_p is the number of future actions. We seek to learn an encoder f_θ that maps the observation sequence $\mathbf{O}_t$ to a latent representation $z = f_\theta(\mathbf{O}_t)$. The learned embedding should exhibit two key properties: (i) observation sequences that lead to behaviorally similar trajectories should be mapped close together in the latent space, and (ii) sequences from behaviorally dissimilar trajectories should be mapped farther apart.

4 Method

In this section, we describe the components of our method. CLASS consists of two stages: (a) pre-computing pairwise action sequence similarities from demonstration data, and (b) representation learning via supervised contrastive learning with a Soft InfoNCE loss.

4.1 Action Sequence Similarity

The first stage of our approach involves measuring similarity between action sequences using Dynamic Time Warping (DTW) [26, 27], which accounts for temporal misalignment between time-series data. Given two action sequences $\mathbf{A}_T^m = \{a_p^m\}_{p=T}^{T+T_p-1}$ and $\mathbf{A}_{T'}^n = \{a_q^n\}_{q=T'}^{T'+T_q-1}$, where a_p^m and a_p^n denote the actions at timesteps p and q from demonstration m and n in the first and second sequences, respectively, the DTW distance is computed as follows:

$$\mathrm{DTW}(\mathbf{A}_T^m, \mathbf{A}_{T'}^n) = \min_{\Gamma} \sum_{(p,q)\in\Gamma} \left\| a_p^m - a_q^n \right\|_2 ,$$

where Γ is a monotonic alignment, and $\| \cdot \|_2$ denotes the Euclidean distance. Pairwise DTW distances are computed across all action sequences in the dataset, and positive pairs are defined as those with distances below a fixed quantile threshold K. The DTW distances of positive pairs are stored for contrastive learning.

4.2 Soft Contrastive Learning

To learn behaviorally informed representation, we use a contrastive learning objective that incorporates *soft positive supervision* using the pre-computed DTW values. Our method resembles that of [28, 29], with modifications for compatibility with policy learning. Specifically, given an input image observation batch, each image

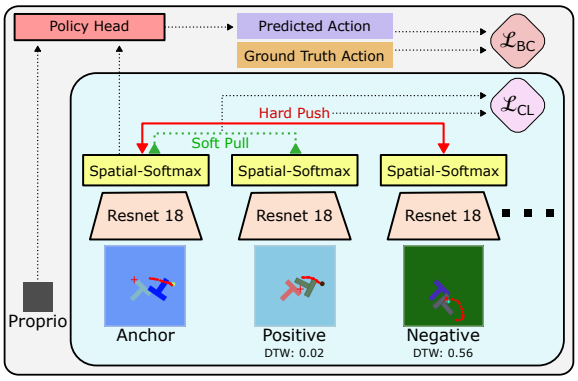


Figure 2: **Training pipeline**. The inner block corresponds to contrastive representation learning as pre-training. The outer block performs policy fine-tuning with proprioception and the vision representation. Modules with the same colors indicate shared weights.

serves as its own anchor, augmented with random cropping and Gaussian noise. During training, the pre-defined positive pairs, identified via DTW, are pulled together, while all other samples are pushed apart. The encoder f_θ is composed of a ResNet-18 [30] backbone and a spatial softmax layer, as shown in Fig. 2. Given an input batch with data augmentation, we encode it into z using f_θ to obtain the L2-normalized image latents $\{\hat{z}_i\}_{i=1}^{B}$, where $\hat{z}_i = z_i/\|z_i\|_2$ and $\|\hat{z}_i\|_2 = 1$. The pairwise similarity matrix $S \in \mathbb{R}^{B\times B}$ is computed as $S_{ij} = \hat{z}_i^\top \hat{z}_j/\tau$ where $\tau > 0$ is a temperature hyperparameter. We define a soft positive weight $w_{ij} \in [0, 1]$ between anchor i and candidate j, computed via DTW-based similarity:

$$w_{ij} = \begin{cases} 1 - \text{CDF}\left(\text{DTW}(i,j)\right), & \text{if } (i,j) \text{ is a positive pair} \\ 0, & \text{otherwise,} \end{cases}$$

where CDF denotes the empirical cumulative distribution function computed over all anchor-positive DTW distances. The soft InfoNCE loss is defined as a weighted log-softmax over positive pairs:

$$\mathcal{L}_{\text{CL}} = -\frac{1}{B}\sum_{i=1}^{B} \frac{\sum_{j=1}^{B} w_{ij} \log p_{ij}}{\sum_{j=1}^{B} w_{ij}}, \quad \text{where} \quad p_{ij} = \frac{\exp(S_{ij})}{\sum_{k=1,\, k\neq i}^{B} \exp(S_{ik})} \tag{1}$$

Eq. (1) generalizes InfoNCE by weighting positives using soft DTW-based scores w_{ij}. This approach enables graded similarity supervision by allowing positive pairs to contribute proportionally to their action sequence similarity. For additional discussion about pre-computation, see Appendix B.

4.3 Training

We use two training schemes. In the first, the encoder is trained solely with the representation learning objective (e.g., the inner block in Fig. 2) without a policy head. In the second scheme, we follow a two-stage protocol: the encoder is first pretrained with the representation objective, and the encoder is fine-tuned with a policy head using the behavior cloning (BC) loss. Additional training details are available in Appendix C.

4.4 Policy Evaluation

We evaluate policies under two different settings. In the **parametric** setting, actions are generated directly by the learned policy network, following standard policy inference procedures. The parametric evaluation requires training a policy head. In the **non-parametric** setting, we extend the approach of Pari et al. [25] to action sequence rollout. At inference time, the most recent observation is encoded into a latent space using the vision encoder, and the k_{nn} nearest neighbors are retrieved from the training data using cosine similarity between representations. Their corresponding action sequences are then ensembled using a similarity-weighted average as follows:

$$\hat{\mathbf{A}} = \frac{\sum_{i=1}^{k_{\text{nn}}} \exp\left(c_i\right) \cdot \mathbf{A}^{(i)}}{\sum_{i=1}^{k_{\text{nn}}} \exp\left(c_i\right)}, \quad \text{with } c_i = \frac{\text{cos_sim}(\hat{z}, \hat{z}^{(i)})}{\tau_{\text{nn}}}$$

where $\hat{\mathbf{A}}$ represents the action sequence rolled out during evaluation, c_i denotes the cosine similarity-based weight, $\hat{z}$ denotes the L2-normalized latent vector obtained by concatenating proprioception and image features, τ_{nn} is a temperature hyperparameter, and $(\hat{z}^{(i)}, \mathbf{A}^{(i)})$ are ith nearest latent observation and action sequence pair in the dataset. The non-parametric setting relies solely on the learned representation and does not require a policy head.

4.5 Theoretical Analysis

The proposed method leverages similarity in action sequences to guide representation learning. As a theoretical justification, we show the relationship between action sequence supervision and the learned latent representation when optimizing the proposed loss function.

Proposition 1. *$\mathcal{L}_{CL}$ optimizes the KL divergence between the distribution of soft positive weights w_{ij} of actions and the distribution of corresponding similarities in the latent representation.*

See Appendix A for the proof. This proposition establishes the relationship between the similarities in the latent representation space and the sequence-level similarities between action trajectories. When the representation is learned, the error is bounded by KL plus a constant factor determined by the quantile threshold we set for the positive pairs.

5 Experiments

We conduct extensive experiments in both simulation and real-world tasks. Our investigation addresses the following questions.

RQ1 Can CLASS learn representations that achieve competitive retrieval-based policy performance in both homogeneous and heterogeneous settings?

RQ2 Does the representation learned with CLASS transfer to downstream policy learning through fine-tuning with BC?

RQ3 How well does CLASS generalize across different settings, including different action spaces and multi-task learning?

5.1 Tasks and Datasets

We evaluate CLASS on 5 simulated and 3 real-robot tasks. Each task represents a class of robotic manipulation tasks. See Appendix D for environment details.

- **Square** is a task from robomimic [31], where the robot places a square nut into a rod, evaluating the basic pick-and-place skill. We use 200 proficient human demonstrations.
- **Three-Stack** is a high-equivariance stacking task from MimicGen [32], where the robot stacks three colored blocks in a specified order. The dataset contains 1000 expert demonstrations with the default reset distribution.
- **Aloha-Transfer** [4] involves a bi-dexterous manipulation task rendered from a top-down view using the Aloha robot. The dataset includes 100 demonstrations. Unlike the other manipulation tasks, it uses bi-robot joint position control, allowing us to assess if CLASS generalizes to different action spaces.
- **LIBERO-Object** [33] is a multi-task benchmark consisting of ten object-centric pick-and-place tasks, each paired with a natural language instruction encoded using a pretrained BERT [34]. Each task includes 50 human-teleoperated demonstrations.
- **Push-T** [3] requires the robot to push a T-shaped object into a designated goal on a 2D plane, evaluating a contact-rich setting. A rollout is considered successful if the intersection-over-union between the object and the goal region exceeds 90%. The dataset has 206 human demonstrations.
- **Two-Stack** is a real-world equivariant task where a red cube is stacked on top of a green cube. 200 demonstrations are collected.
- **Mug-Hang** is a real-world precise placement task where the robot picks up a mug placed at a random pose and hangs it onto a mug rack. 200 demonstrations are collected.
- **Toaster-Load** is a real-world task where the robot picks up a piece of soft bread and places it into a toaster slot at a random pose. 200 demonstrations are collected.

5.2 Baselines

We follow the training and evaluation procedures described in Sections 4.3 and 4.4. Baselines are categorized into two groups: representation learning methods for pre-training the vision encoder, and policy classes for fine-tuning with different policy head architectures and BC objectives. We include results for different combinations of the two groups and report them in Table 1.

Representation Learning

- **Random, ImageNet, R3M**: ResNet-18 trained from scratch, or initialized with ImageNet-1K [35] or R3M [17] pre-trained weights.

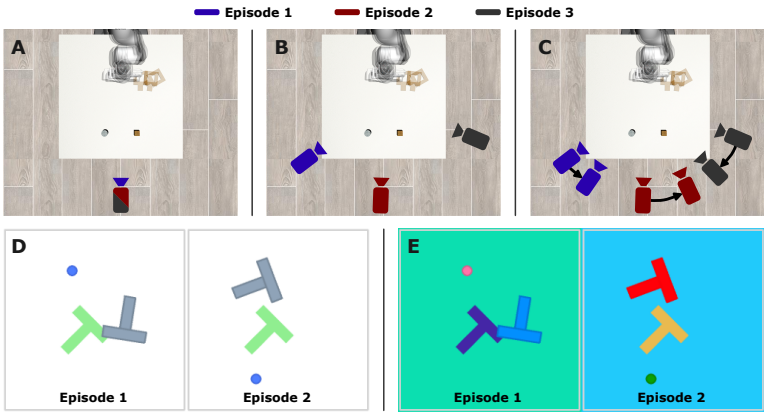


Figure 3: **Heterogeneous data collection setups.** **(A)** Fixed camera (**Fixed-Cam**), commonly used in conventional behavior cloning pipelines. **(B)** Random static camera (**Rand-Cam**), where the camera pose is randomly sampled at the start of each episode but remains fixed during the episode. **(C)** Dynamic camera (**Dyn-Cam**), where a randomly initialized camera moves during the episode with a random direction while maintaining a consistent look-at target. **(D)** Fixed object color (**Fixed-Color**), commonly assumed in vision-based behavior cloning tasks. **(E)** Random object color (**Rand-Color**), where the color of the objects is randomly varied in each demonstration. We compare setups (A) and (C) for simulated tasks, (D) and (E) for the Push-T task, and evaluate under setup (B) for real-world tasks. See Appendix E for additional details.

- **TCN**: A single-view time-contrastive network [16].
- **VINN**: A Bootstrap Your Own Latent (BYOL) [36] pre-training on demonstrations [25].
- **DynaMo**: In-domain pre-training with inverse and forward dynamics model [37].
- **EquiVar**: Rotation group-equivariant encoder that improves data efficiency for DP [38].

Policy Class When fine-tuning with the BC objective, we consider two policy classes:

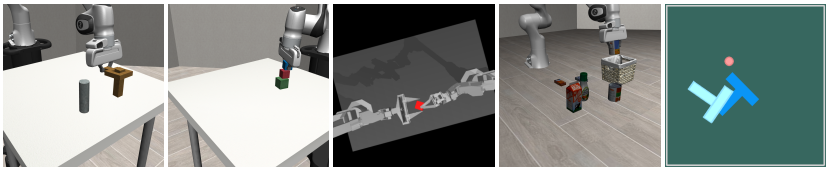
- **MLP**: A CNN-based policy head with a direct supervised learning objective.
- **DP** (Diffusion Policy): A CNN-based policy head trained with diffusion objective [3].

5.3 Heterogeneous Data Collection Setup

Behavior cloning methods are commonly trained on demonstrations collected from a consistent viewpoint, often using a static, fixed camera—a condition that humans do not require. While this assumption simplifies the learning problem by enforcing visual consistency, it breaks down at scale, where data naturally spans diverse camera setups and collection conditions [2]. Even minor variations in camera pose between demonstrations have been shown to substantially degrade performance [6, 7]. To evaluate generalization under such visual shifts, we construct settings where each demonstration begins from a randomized camera pose, with optional movements during the episode to introduce additional viewpoint variation. For the Push-T task, we randomize object and background colors to test robustness to appearance shifts, as shown in Fig. 3. To evaluate compositional generalization across demonstrations, we do not apply rotation or color augmentations during training. For additional details, see Appendix E.

5.4 Simulation Results

Table 1 summarizes the simulation performance of CLASS compared to baseline methods. CLASS consistently improves success rates across all tasks and policy classes: Across the static and dynamic settings, it achieves **85%** success rate with **MLP** and **91%** with **Diffusion Policy (DP)** on average, significantly outperforming the best baselines, which reach only 63% and 77%, respectively. Larger gains are observed under dynamic camera and random object color setups, where CLASS achieves mean success rates of 76% (MLP) and 85% (DP), compared to 32% and 57% from the best baselines. In the non-parametric (Rep-Only) setting, CLASS achieves success rates comparable to parametric DP, despite using no policy head, with only a 9% drop in mean success rate with an average of 83%



	Square		Three-Stack		Aloha-Transfer		LIBERO-Object		Push-T	
	Fixed-Cam	Dyn-Cam	Fixed-Cam	Dyn-Cam	Fixed-Cam	Dyn-Cam	Fixed-Cam	Dyn-Cam	Fixed-Color	Rand-Color
Rep-Only										
ImageNet	0.20	0.00	0.00	0.00	0.26	0.02	0.82	0.59	0.00	0.00
R3M	0.04	0.00	0.02	0.00	0.28	0.00	0.57	0.56	0.00	0.00
TCN	0.53	0.07	0.28	0.01	0.17	0.00	0.90	0.92	0.04	0.02
VINN	0.13	0.04	0.00	0.00	0.13	0.02	0.92	0.62	0.00	0.01
DynaMo	0.01	0.00	0.00	0.00	0.11	0.09	0.58	0.57	0.00	0.02
CLASS	**0.93**	**0.70**	**0.94**	**0.77**	**0.91**	**0.56**	**0.99**	**0.99**	**0.85**	**0.66**
MLP										
ImageNet	0.61 / 0.91	0.11 / 0.18	0.31 / 0.91	0.05 / 0.36	0.25 / **1.00**	0.10 / 0.33	0.99 / **1.00**	0.60 / 0.46	0.66 / 0.87	0.21 / 0.27
CLASS	**0.88 / 0.95**	**0.67 / 0.62**	**0.62 / 0.93**	**0.61 / 0.80**	**0.72 / 1.00**	**0.52 / 0.78**	**1.00** / 0.99	**1.00 / 0.96**	**0.82 / 0.89**	**0.69 / 0.64**
DP										
Random	0.45 / 0.85	0.06 / 0.06	0.53 / 0.86	0.05 / 0.38	0.32 / **1.00**	0.11 / 0.15	0.99 / 0.91	0.57 / 0.04	0.62 / 0.89	0.04 / 0.16
ImageNet	0.72 / 0.91	0.09 / 0.21	0.86 / 0.94	0.07 / 0.61	0.37 / **1.00**	0.09 / 0.20	**1.00** / 0.97	0.64 / 0.19	0.70 / 0.92	0.15 / 0.53
R3M	0.42 / 0.88	0.04 / 0.14	0.36 / 0.84	0.02 / 0.36	0.35 / **1.00**	0.10 / 0.09	0.99 / 0.94	0.60 / 0.12	0.65 / 0.89	0.00 / 0.05
TCN	0.78 / 0.91	0.17 / 0.34	0.85 / 0.93	0.12 / 0.71	0.52 / **1.00**	0.11 / 0.33	0.98 / **0.99**	0.95 / 0.96	0.69 / 0.96	0.33 / 0.49
VINN	0.57 / 0.87	0.07 / 0.07	0.63 / 0.88	0.01 / 0.22	0.39 / **1.00**	0.08 / 0.15	0.92 / **0.99**	0.57 / 0.06	0.66 / 0.93	0.13 / 0.06
DynaMo	0.39 / 0.80	0.07 / 0.05	0.48 / 0.81	0.02 / 0.28	0.82 / 0.88	0.17 / 0.23	0.96 / **0.99**	0.46 / 0.40	0.67 / 0.90	0.06 / 0.24
EquiVar	0.77 / 0.94	0.03 / 0.08	0.43 / 0.95	0.06 / 0.36	–	–	0.99 / **0.99**	0.61 / 0.10	–	–
CLASS	**0.91 / 0.95**	**0.64 / 0.68**	**0.89 / 0.97**	**0.76 / 0.93**	**0.83** / 0.99	**0.61 / 0.95**	**1.00 / 0.99**	**1.00 / 0.97**	**0.81 / 0.96**	**0.69 / 0.70**

Table 1: **Performance comparison between CLASS and baselines on simulation tasks.** We report the maximum task success rate across 50 randomly initialized scenes except for LIBERO-object which tests each task 20 times, averaged over three seeds. All methods are evaluated ten times throughout training at regular intervals. Each cell shows **non-parametric / parametric** evaluation when applicable. For Rep-Only settings, only non-parametric results are reported. All results use a single global-view camera. Whenever applicable, ImageNet pre-training weights are used for representation learning. For additional discussions and evaluations, see Appendix F.

across all tasks. Finally, strong performance on Aloha-Transfer and LIBERO-Object demonstrates that CLASS generalizes effectively to different action spaces and multi-task learning scenarios.

5.5 Design Choice Analysis for CLASS

We evaluate the impact of key design choices in CLASS, including soft contrastive learning, the use of a sequence-level similarity metric, the choice of DTW for measuring action similarity, and the positive pair selection threshold. Fig. 4(a) compares soft contrastive learning against hard contrastive learning, where the soft positive weights $w_{ij} > 0$ in Equation 1 are replaced with uniform weights of 1. Critically, it is shown that removing the graded similarity signals leads to significant

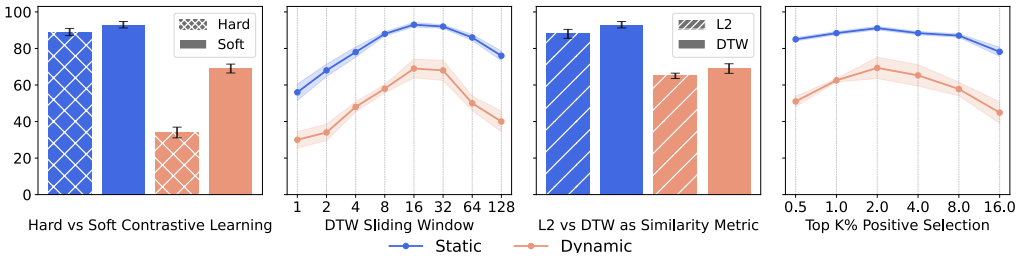


Figure 4: **Design Choice Analysis**: Success rate of Fixed-Cam (blue) and Dynamic-Cam (orange) for Square task with Rep-Only as a function of (a) hard or soft contrastive learning, (b) size of DTW sliding window, (c) similarity metric, and (d) the quantile K (%) value for positive sample selection.

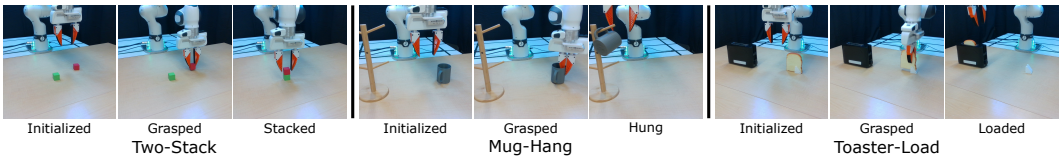


Figure 6: **Real-world task execution.** The initial scenes and the subtasks are visualized.

	Two-Stack (real)		**Mug-Hang** (real)		**Toaster-Load** (real)	
Method	Grasped	Stacked	Grasped	Hung	Grasped	Loaded
ImageNet-DP	0.15 / 0.30	0.00 / 0.10	0.15 / 0.05	0.00 / 0.00	0.95 / **1.00**	0.00 / 0.05
CLASS-DP	**0.70 / 0.80**	**0.45 / 0.60**	**0.75 / 0.80**	**0.55 / 0.65**	**1.00 / 1.00**	**0.35 / 0.55**

Table 2: The performance of CLASS pre-trained DP for real-world tasks in comparison to ImageNet pre-trained DP. We report the success rate for each subtask over 20 randomly initialized scenes and camera placements. Each cell shows the results using the non-parametric / parametric evaluations.

degradation of performance. Fig. 4(b) shows that increasing the sliding window size for calculating DTW between action sequences improves success rates up to $T = 16$, indicating the importance of using sequence-level similarity. As shown in Fig. 4(c), DTW outperforms L2 distance in both static and dynamic settings. Finally, varying the positive selection quantile in Fig. 4(d) reveals a tradeoff: small thresholds limit available information, while large thresholds lead to more false positives. See Appendix C for implementation details.

5.6 Real-World Experiment / Results

We design three real-world manipulation tasks as described in Section 5.1, using a Franka Research 3 robot and a single RealSense D435 camera capturing 720p RGB images at 30 FPS. The images are cropped and resized to $256 \times 256 \times 3$. A proficient expert teleoperates the robot using a VR controller. To introduce visual variation, the camera is mounted on a tripod and randomly repositioned after each demonstration, as illustrated in Fig. 5. For evaluation, we compare Diffusion Policy pre-trained on ImageNet (ImageNet-DP) with CLASS-DP, which uses representations learned via the CLASS objective. All results use a single global-view camera.

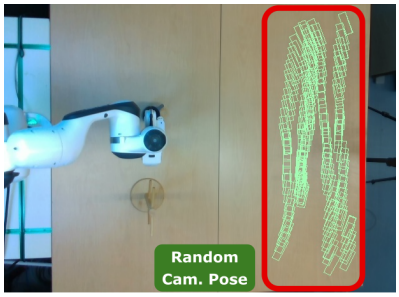


Figure 5: Visualization of camera poses for the Mug-Hang task.

Table 2 summarizes performance on real-world tasks. CLASS-DP outperforms ImageNet-DP across all three tasks, achieving on average **37%** and **45%** higher subtask success and final task completion rates, respectively, in the non-parametric setting. In the parametric setting, the differences increase to **41%** and **55%**, respectively. These results demonstrate that CLASS generalizes effectively to real-world settings and improves policy performance when learning from demonstrations with varying viewpoints.

6 Conclusion

This paper presents CLASS, a novel framework for imitation learning in robotic manipulation based on soft contrastive learning with an action sequence similarity metric. We confirm that BC suffers from its strong end-to-end supervision mechanism, especially when learning from heterogeneous datasets. In contrast, CLASS learns representations that map observations with similar trajectory-level behaviors closer in feature space, enabling robust representation learning in both homogeneous and heterogeneous settings. Across 5 simulated and 3 real-world tasks, CLASS consistently outperforms baselines in both retrieval-based evaluations and downstream policy fine-tuning. Future directions include extending CLASS to large-scale policy learning and cross-embodiment learning with a unified action space, leveraging diverse offline data to enable few-shot learning, and investigating its effectiveness when learning from suboptimal or noisy demonstrations.

Limitations

While CLASS demonstrates strong performance across a range of tasks and settings, several limitations remain. First, CLASS requires additional pre-computing pairwise DTW distances to define positive pairs, which has a quadratic cost in the number of data points. However, CLASS's faster convergence during training can make it more computationally efficient in repeated training runs. For more detailed analysis, see Appendix G. Second, the use of suboptimal or noisy demonstrations is not addressed in this work. Lastly, the current scope of the work is limited to vision modality.

Failure modes On real-world tasks, the most common failure mode involves the robot proceeding to the next subtask region without completing the current one. For example, in the Mug-Hang task, the policy occasionally fails to grasp the mug but still moves toward the hanger with the gripper closed. During non-parametric evaluations, we observe less fine-grained control, particularly on high-precision tasks such as Toaster-Load, where it fails to accurately insert the bread into the slot. Additionally, when the camera is positioned outside the pre-defined region used during data collection, performance degrades significantly, indicating that the policy still has a limited extrapolative generalization capability to unseen viewpoints.

Acknowledgment

This research was partly supported by Delta Electronics Inc., Toyota Research Institute, and NSF CMMI-2443076. We acknowledge Research Computing at the University of Virginia for providing the computational resources that made the results in this work possible.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[2] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[3] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[4] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[5] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[6] A. Xie, L. Lee, T. Xiao, and C. Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3153–3160. IEEE, 2024.

[7] V. Saxena, M. Bronars, N. R. Arachchige, K. Wang, W. C. Shin, S. Nasiriany, A. Mandlekar, and D. Xu. What matters in learning from large-scale datasets for robot manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[8] Z. Li, R. Pérez-Dattari, R. Babuska, C. Della Santina, and J. Kober. Beyond behavior cloning: Robustness through interactive imitation and contrastive learning. *arXiv preprint arXiv:2502.07645*, 2025.

[9] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 1273–1286. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/099fe6b0b444c23836c4a5d07346082b-Paper.pdf.

[10] S.-W. Lee, X. Kang, and Y.-L. Kuo. Diff-dagger: Uncertainty estimation with diffusion policy for robotic manipulation. In *International Conference on Robotics and Automation (ICRA)*, 2025.

[11] L. Lai, A. Z. Huang, and S. J. Gershman. Action chunking as policy compression. 2022.

[12] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[13] A. van den Oord, O. Vinyals, and k. kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf.

[14] N. M. M. Shafiullah, Z. J. Cui, A. Altanzaya, and L. Pinto. Behavior transformers: Cloning k modes with one stone. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=agTr-vRQsa.

[15] B. Eysenbach, T. Zhang, S. Levine, and R. R. Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 35603–35620, 2022.

[16] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[17] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[18] Y. J. Ma, S. Sodhani, D. Jayaraman, O. Bastani, V. Kumar, and A. Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations*, 2023.

[19] M. Halawa, O. Hellwich, and P. Bideau. Action-based contrastive learning for trajectory prediction. In *European conference on computer vision*, pages 143–159. Springer, 2022.

[20] Q. Zhang, Z. Peng, and B. Zhou. Learning to drive by watching youtube videos: Action-conditioned contrastive policy pretraining. In *European Conference on Computer Vision*, pages 111–128. Springer, 2022.

[21] H. Zhuang, W. Emma Zhang, J. Yang, W. Chen, and Q. Z. Sheng. Not all negatives are equally negative: Soft contrastive learning for unsupervised sentence representations. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 3591–3601, 2024.

[22] V. Suresh and D. C. Ong. Not all negatives are equal: Label-aware contrastive loss for fine-grained text classification. *arXiv preprint arXiv:2109.05427*, 2021.

[23] H. Li, X. Zhou, L. A. Tuan, and C. Miao. Rethinking negative pairs in code search. *arXiv preprint arXiv:2310.08069*, 2023.

[24] S. Lee, T. Park, and K. Lee. Soft contrastive learning for time series. *arXiv preprint arXiv:2312.16424*, 2023.

[25] J. Pari, N. M. Shafiullah, S. P. Arunachalam, and L. Pinto. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021.

[26] M. Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[27] M. Middlehurst, A. Ismail-Fawaz, A. Guillaume, C. Holder, D. Guijo-Rubio, G. Bulatova, L. Tsaprounis, L. Mentel, M. Walter, P. Schäfer, and A. Bagnall. aeon: a python toolkit for learning from time series. *Journal of Machine Learning Research*, 25(289):1–10, 2024. URL http://jmlr.org/papers/v25/23-1444.html.

[28] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

[29] C. Ge, J. Wang, Z. Tong, S. Chen, Y. Song, and P. Luo. Soft neighbors are positive supporters in contrastive visual representation learning. *arXiv preprint arXiv:2303.17142*, 2023.

[30] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[31] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[32] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

[33] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[36] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33: 21271–21284, 2020.

[37] Z. Cui, H. Pan, A. Iyer, S. Haldar, and L. Pinto. Dynamo: In-domain dynamics pretraining for visuo-motor control. *Advances in Neural Information Processing Systems*, 37:33933–33961, 2024.

[38] D. Wang, S. Hart, D. Surovik, T. Kelestemur, H. Huang, H. Zhao, M. Yeatman, J. Wang, R. Walters, and R. Platt. Equivariant diffusion policy. *arXiv preprint arXiv:2407.01812*, 2024.

[39] Y. You, I. Gitman, and B. Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

[40] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[41] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5745–5753, 2019.

[42] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

A Proof of Proposition 1

Let $q_{ij} = w_{ij}/Z$ where $Z = \sum_{j=1}^{B} w_{ij}$ is the partition function. We can write the instance-wise proposed loss as follows:

$$l_{\text{CL}}^{(i)} = -\frac{\sum_{j=1}^{B} w_{ij} \log p_{ij}}{\sum_{j=1}^{B} w_{ij}} = -\sum_{j=1}^{B} \frac{w_{ij}}{Z} \log p_{ij} = \sum_{j=1}^{B} q_{ij} \log \frac{q_{ij}}{p_{ij}} - \underbrace{q_{ij} \log q_{ij}}_{\text{constant}}$$

Let P and Q be the probability distribution of p_{ij} and q_{ij}. We can rewrite the instance-wise loss as

$$l_{\text{CL}}^{(i)} = D_{\text{KL}}(Q||P) + \text{const}$$

This is the KL divergence of the predicted softmax similarity from the soft positive assignments.

B DTW-based Similarity Calculation

To compute DTW between action sequences, we use `aeon` [27], a Python toolkit that supports efficient DTW computation for multivariate time-series data. For the Aloha-Transfer environment, each action sequence is 8-dimensional, comprising 7D joint positions and a 1D gripper action. In Push-T, actions are 2-dimensional, controlling the absolute 2D target position. For the other environments that use end-effector pose control, each action sequence is 10-dimensional, consisting of 3D position, 6D orientation, and 1D gripper action. To balance the influence of position and orientation, orientation values are scaled by a factor of 0.5 prior to the DTW calculations.

To enable consistent interpretation of DTW distances, we modeled the empirical cumulative distribution function (CDF) from a set of pairwise DTW values. In our experiments, we precomputed these distances for all anchor-positive pairs. As noted in the main text, this approach is more efficient than constructing a full $N \times N$ distance matrix and was feasible for the datasets used.

However, we recognize that this pre-computation method is not fully scalable. While it avoids the full distance matrix, the computational and memory cost still scales quadratically with the number of anchor-positive pairs, becoming prohibitive for large datasets. For these scenarios, an online DTW computation strategy offers a potential alternative. This method would estimate the CDF from a data sample and compute pairwise distances on-the-fly within training batches. This bypasses the need for pre-computation and large-scale storage, as the computational overhead is tied to the much smaller batch size, thereby extending the scalability of our method to larger datasets.

C Implementation Details and Hyperparameters

C.1 Representation Learning

We focus on training the vision encoder for representation learning. Except for Equivariant Diffusion Policy, all methods use a ResNet-18 backbone followed by a spatial softmax layer. All batch normalization layers are replaced with group normalization layers.

Name	Value
Gradient clip norm	0.5
Scheduler	Cosine
Warmup steps	500
Use EMA	True
EMA Power	0.75

Table 3: Shared hyperparameters for representation learning methods.

Contrastive Learning via Action Sequence Supervision (CLASS)
Table 4 lists the shared training and evaluation hyperparameters used across all tasks. T_a indicates

the rollout horizon for action sequences during evaluation. Table 5 shows task-specific settings for Top K (%) similarity thresholds for positive sample selection and number of pre-training epochs. Epochs are tuned based on dataset size and validation performance.

Name	**Value**
Pre-computing	
DTW Horizon	16
Training	
Optimizer	LARS [39]
Batch size	160
Learning rate	0.5
Momentum	0.9
Weight decay	$1e^{-6}$
τ	0.05
Non-parametric Evaluation	
T_a	16
k_{nn}	64

Table 4: Shared hyperparameters for CLASS.

Task	**Top K (%)**	**Epochs**	τ_{nn}
Square	2.5	50	0.01 / 0.02
Three-Stack	1.0	10	0.01 / 0.02
Aloha-Transfer	2.5	50	0.01 / 0.02
LIBERO-Object	2.5	50	0.01 / 0.02
Push-T	1.5	100	0.001 / 0.002
Two-Stack	2.5	50	0.02
Mug-Hang	2.5	50	0.02
Toaster-Load	2.5	50	0.02

Table 5: Task-specific hyperparameters for CLASS. Homogeneous and heterogeneous settings are distinguished in the τ_{nn} values in each cell.

Time Contrastive Network (TCN)
We use the single-view TCN implementation from https://github.com/kekeblom/tcn. The model is trained for 200 epochs with a batch size of 64 and a learning rate of $1e^{-4}$, where each batch consists of 64 triplets of images used for contrastive learning.

Visual Imitation through Nearest Neighbors (VINN)
We follow the BYOL implementation from https://github.com/lucidrains/byol-pytorch. The model is trained for 200 epochs with a batch size of 64 and a learning rate of $1e^{-4}$.

Dynamics pre-training for Visuo-Motor Control (DynaMo)
We follow the procedures from the original paper. The encoder is based on a ResNet-18 backbone and is trained for 40 epochs with a batch size of 64 and a learning rate of $1e^{-4}$.

C.2 Policy Learning

For all experiments, we choose reduced the learning rate for the vision encoder to better preserve the pre-trained representation. The learning rate for the vision encoder is $1e^{-5}$ and the learning rate for the policy head is $1e^{-4}$.

MLP
The architecture is based on a U-Net structure adapted from the CNN-based Diffusion Policy (DP),

Name	Value
Epochs	40
Batch size	64
Obs. context	5
Forward dynamics dropout	0.0
Transition latent dim	32

Table 6: Hyperparameters for DynaMo.

modified to support action chunking with direct action sequence supervision. Table 7 includes the policy hyperparameters.

Diffusion Policy
We adopt the CNN-based model architecture from Chi et al. [3]. Table 8 includes diffusion-specific hyperparameters for regular DP and Equivariant DP.

Equivariant Diffusion Policy (EquiVar)
We focus on the equivariant vision encoder as a replacement of the ResNet-18 encoder, following the official implementation from https://github.com/pointW/equidiff. The vision encoder is trained from scratch for policy training. The model architecture is modified to accept input images of size $224 \times 224 \times 3$.

Name	Value
Training	
Batch size	64
Weight decay	$1e^{-6}$
Betas	(0.9, 0.99)
Gradient clip norm	0.5
Scheduler	Cosine
Warmup steps	500
Use EMA	True
EMA Power	0.75
T_o	1
T_p	16
Parametric Evaluation	
T_a	12

Table 7: Shared policy hyperparameters for MLP and DP.

Name	Value
Training	
Model Type	DDIM [40]
Prediction Type	epsilon
Denoising Step	16
Parametric Evaluation	
Denoising Step	16

Table 8: Diffusion-specific hyperparameters.

Task	Training Epochs	Fine-tuning Epochs
Square	500	100
Three-Stack	100	20
Aloha-Transfer	500	400
LIBERO-Object	500	100
Push-T	500	250
Two-Stack	500	100
Mug-Hang	500	100
Toaster-Load	500	100

Table 9: Task-specific hyperparameters for policy head training. Training Epochs denote the number of epochs used to train the BC policy head for the baseline methods. Fine-tuning Epochs indicate the number of epochs used to fine-tune the policy head after CLASS pre-training. CLASS pre-training enables faster convergence speed during fine-tuning and thus requires less number of epochs.

D Environments Details

All environments use position-based control. For tasks involving control of the end-effector pose, orientation is represented using the continuous 6D representation proposed by Zhou et al. [41].

Task Name	Observation Modalities	Image Resolution	Action Space	Max Steps
Simulation				
Square	Agent-View Image, End-Effector Pose, Joint Position, Gripper State	$256 \times 256 \times 3$	End-Effector Position	400
Three-Stack	Agent-View Image, End-Effector Pose, Joint Position, Gripper State	$256 \times 256 \times 3$	End-Effector Position	600
Aloha-Transfer	Top-Down Image, Joint Position, Gripper State	$256 \times 256 \times 3$	Bi-manual Joint Position	600
Libero-Object	Agent-View Image, End-Effector Pose, Joint Position, Gripper State	$256 \times 256 \times 3$	End-Effector Position	400
Push-T	Top-Down Image, Agent Position	$256 \times 256 \times 3$	2D Position	500
Real-world				
Two-Stack	Agent-View Image, End-Effector Pose, Joint Position, Gripper State	$256 \times 256 \times 3$	End-Effector Position	600
Mug-Hang	Agent-View Image, End-Effector Pose, Joint Position, Gripper State	$256 \times 256 \times 3$	End-Effector Position	600
Toaster-Load	Agent-View Image, End-Effector Pose, Joint Position, Gripper State	$256 \times 256 \times 3$	End-Effector Position	600

Table 10: Observation modalities, image resolution, action space, and maximum episode length for each task.

E Heterogeneous Data Illustration

E.1 Simulation Camera Placement

For LIBERO-Object, Square, and Three-Stack tasks, we implement a moving camera by attaching it to a mocap body and updating its pose at every environment step. The camera is initialized at a random position on a sphere of radius 1.0 centered at a fixed look-at target. The azimuth θ is sampled from $[0, 2\pi)$, and the elevation ϕ from $[\pi/3, 2\pi/3]$, ensuring relatively horizontal views. The orientation is set to point at the target using a look-at transform with world up-vector $[0, 0, 1]$. At the start of each episode, a direction vector is sampled from $x \in [-1, 1]$, $y \in [-0.01, 0.3]$, then projected onto the tangent plane at the initial camera position. The camera moves along this direction by rotating the position vector around the perpendicular axis, with a fixed arc step of $1.5e^{-4}$ per timestep. The orientation is updated each step to remain focused on the target.

For the ALOHA-TRANSFER task, we simulate camera motion by introducing a consistent per-episode image rotation on the top-down view. At the start of each episode, an initial rotation angle is uniformly sampled from $[0°, 360°)$, and a fixed rotation direction of either $+0.5°$ or $-0.5°$ per timestep is chosen. At each step, the image is rotated accordingly.

E.2 Object/Background Color

For the PUSH-T task, we introduce per-episode color variation by randomly sampling RGB values for the background, goal region, tee block, and agent. The background color is constrained to avoid

strong red tones. To ensure visibility, the goal, tee, and agent colors are sampled to be visually distinct from the background and from each other, based on a minimum Euclidean distance in RGB space. Colors remain fixed throughout each episode but vary across episodes.

E.3 Real-world Camera Placement

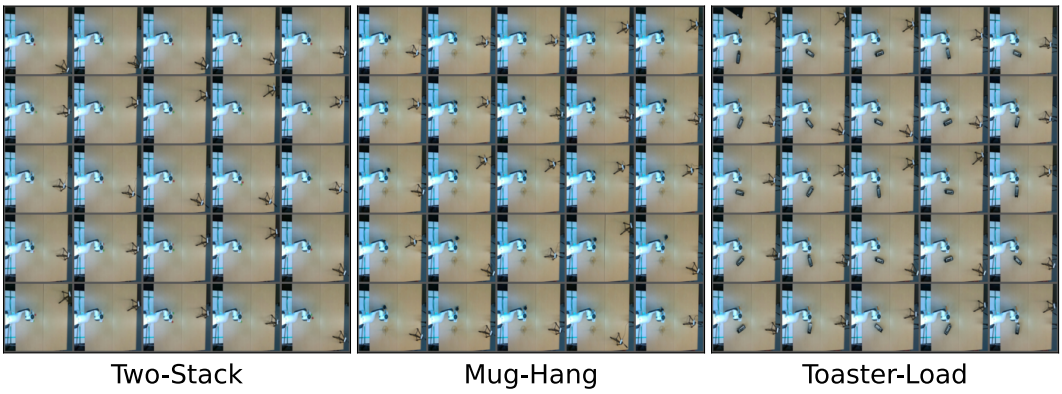


Figure 7: Top-down views from 25 randomly selected episodes, illustrating the distribution of real-world camera placements across the three real-world tasks.

Figure 7 shows top-down views from 25 randomly sampled episodes across the three real-world tasks. The variation in camera placement reflects the heterogeneity in data collection setups.

F Additional Experiments

To evaluate the compositionality of the learned representation and to challenge methods to generalize without additional observations—such as a wrist-view camera—we constrain our original experimental setup to a single global view camera and apply image augmentations excluding color and rotational transformations. In this section, we demonstrate that incorporating a wrist-view camera or applying additional task-aligned data augmentations—both expected to benefit Behavior Cloning (BC)—also enhances CLASS performance. Crucially, CLASS still retains its advantages over BC under these conditions.

F.1 Additional Wrist-View Camera

In Figure 9(a), we evaluate adding a wrist-mounted camera for the Dynamic Camera Square task. The wrist camera provides a consistent viewpoint that is invariant to the global camera placement variations tested in our main study, which is expected to boost performance for behavior cloning.

With the wrist camera view added, the performance of fine-tuned policy with CLASS pre-training increases the from 68% to 90% success rate, while BC improves from 21% to 66%. This demonstrates that while providing a stable, ego-centric view does help both methods, CLASS still significantly outperforms BC.

F.2 Additional Color Augmentation

Figure 9(b) examines color augmentation on the Rand-Color Push-T task, where object colors are randomized to test visual robustness. We apply random hue, saturation, and brightness shifts during

training, which is expected to improve generalization by reducing overfitting to specific color patterns. With color augmentation, the fine-tuned policy with CLASS pre-training increases from 72% to 90% success rate, while BC improves from 53% to 78%. This confirms that while data augmentation benefits both approaches, CLASS pre-training continues to provide substantial performance gains even when visual robustness is enhanced through augmentation.

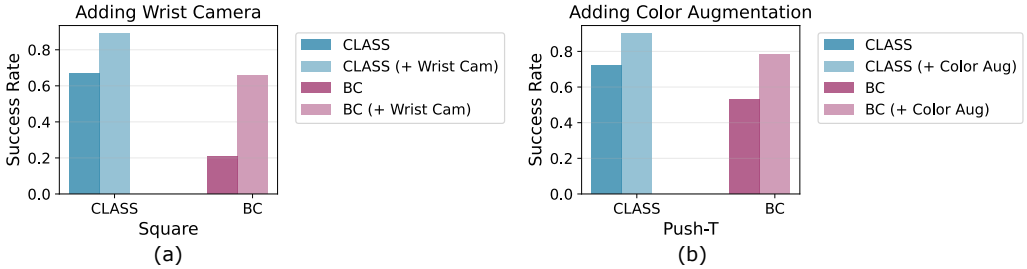


Figure 8: Impact of adding wrist camera and task-aligned augmentations. (a) Success rates of the Dynamic Camera Square task with and without adding the wrist camera. (b) Success rates of the Random Color Push-T task with and without color augmentation during training. Both experiments are reported using parametric evaluation with BC fine-tuned policies with and without CLASS pre-training.

F.3 Benefits of ImageNet Pre-training for CLASS

In our paper, all representation methods utilize ImageNet pre-training whenever applicable. In this section, we show the benefits of using ImageNet pre-trained weights for CLASS. We conduct an ablation study where we compare the policy performance using CLASS pre-training with and without ImageNet pre-trained Resnet-18 as initial weights for Square, Three-Stack, and Push-T tasks. As shown in Figure 9, training from scratch leads to significant drops in success rates across all three tasks. The performance drop is more pronounced in heterogeneous settings, including both Dynamic Camera and Random Color settings. Most significantly, training from scratch for Random Color Push-T task leads to 64% drop in performance in comparison to using the ImageNet pre-trained weights.

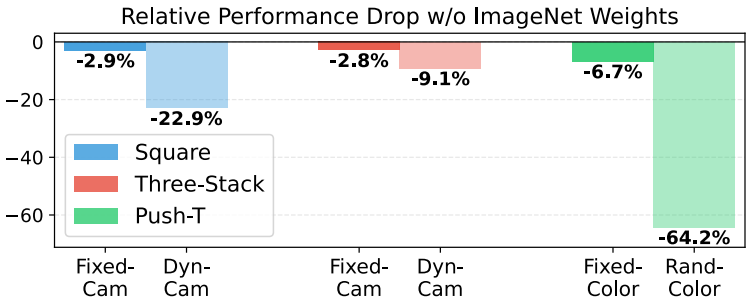


Figure 9: The relative performance drop when training from scratch instead of using ImageNet pre-trained weights for CLASS pret-raining. The results are reported using non-parametric rollouts, averaged over three seeds.

G Training Dynamics

Although pre-computation introduces additional computational overheads, CLASS benefits from fast convergence during both pre-training and fine-tuning. CLASS pre-training converges much faster than BC training from scratch, as shown in Figure 10 across different tasks. In addition, as shown in Table 9, CLASS works with a fewer number of epochs during BC fine-tuning due to fast

convergence rate for the policy learning, thanks to action-centric pre-training. This means that in repeated training runs, CLASS becomes more computationally efficient than BC, where there are multiple pre-training or fine-tuning attempts following a one-time pre-computation.

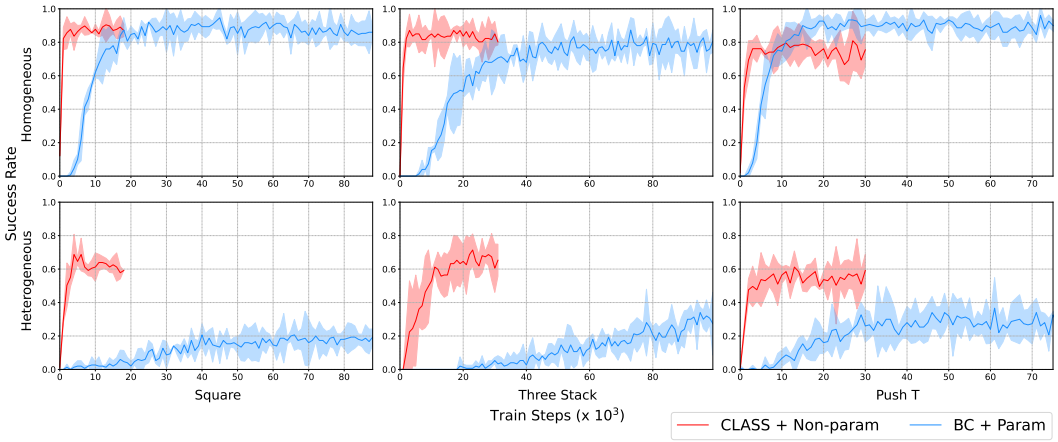


Figure 10: Success Rate during Training for Dynamic Cam Square, Dynamic Cam Three Stack, and Random Color Push-T Rasks. In this plot, CLASS utilizes non-parametric rollouts using the learned representation, whereas BC uses diffusion policy head with parametric rollouts. CLASS requires significantly less number of training steps than BC and is more compute-efficient in repeated training runs.

H Scaling Law

Similar to BC, CLASS follows a scaling law with improved performance as the dataset size increases. Larger datasets yield higher-quality and behaviorally-relevant positive pairs, thereby improving the contrastive signal and learned representations. To illustrate this trend, we evaluate policy performance using 20, 50, 100, 200, 500, and 1000 demonstrations on the Three-Stack task, as shown in Fig. 11. CLASS with non-parametric rollouts consistently outperforms BC with parametric rollouts across different dataset sizes for both fixed and dynamic camera settings.

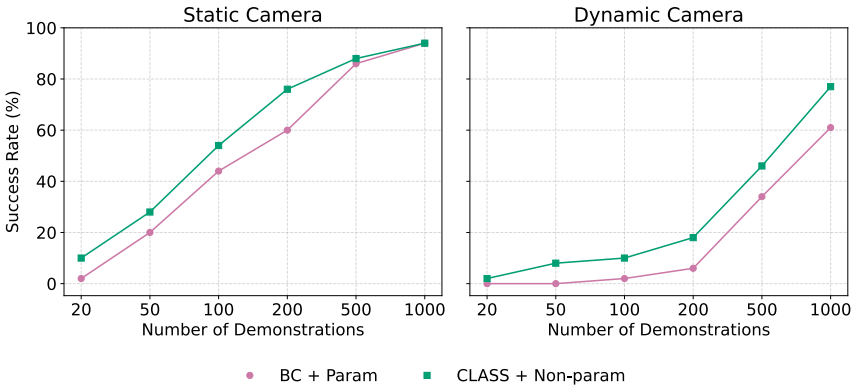


Figure 11: Success rates on the Three-Stack task under varying dataset sizes (20–1000 demonstrations). We compare CLASS pre-training and behavioral cloning (BC) from scratch under both fixed and dynamic camera settings. CLASS consistently outperforms BC across all data regimes, demonstrating its scalability with larger datasets.

I Inference Time

Inference efficiency varies across policy classes. Representation-only methods avoid the overhead of a policy head and rely solely on embedding retrieval, resulting in fast inference, especially for small datasets. This makes them well-suited for high-frequency control. However, their inference time scales with dataset size due to retrieval costs. We compare inference times across policy classes to show these differences in Figure I. On average, Rep-Only methods achieve an inference time of 5.5 ms, compared to 7.3 ms for MLP and 84.4 ms for DP.

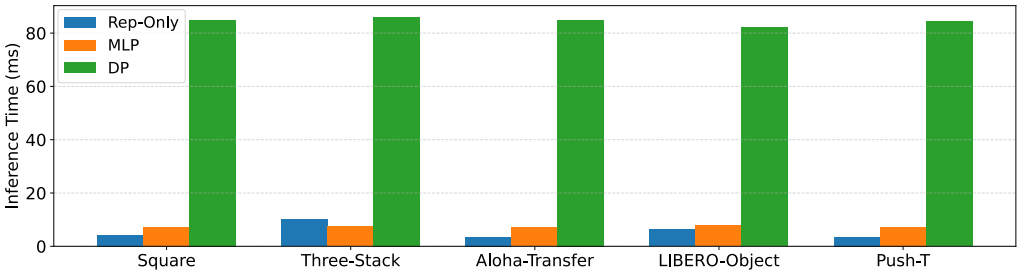


Figure 12: Inference time across policy classes and simulation tasks, averaged over 100 runs on an NVIDIA A40 GPU. Rep-Only methods are fastest on average but scale with dataset size due to the retrieval process. MLP and DP have fixed inference costs, with DP being slower due to its iterative denoising process. The dataset sizes for Square, Three-Stack, Aloha-Transfer, LIBERO-Object, and Push-T are 30K, 254K, 40K, 75K, and 27K samples, respectively.

J Visualization of Representation

J.1 Latent Trajectory

We visualize latent trajectories for the Dynamic Camera Square task in Fig. 12 to assess representation quality. Models trained with standard behavioral cloning and with CLASS pre-training are compared. CLASS learns representations where different demonstrations align and overlap in latent space, indicating higher transferability. In contrast, the BC-trained representation shows little overlap across demonstrations, suggesting poor generalization and less structured embeddings.

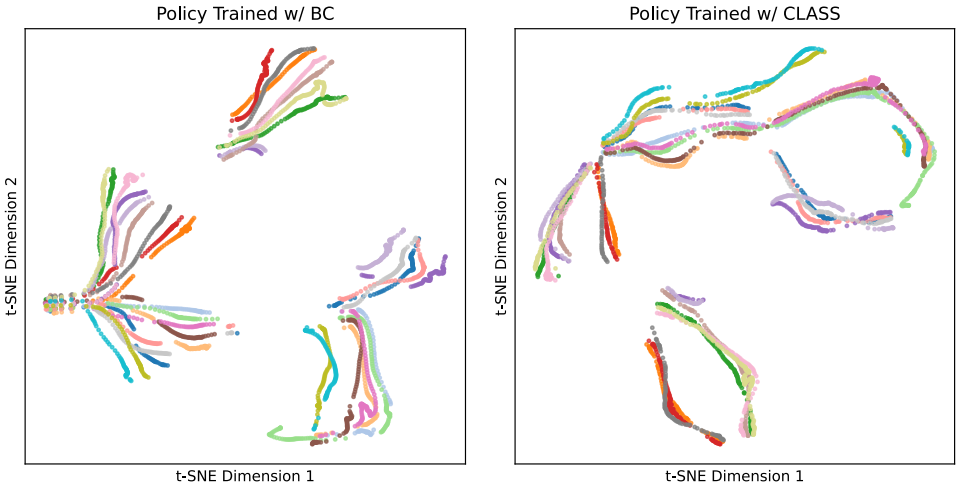


Figure 13: Visualization of latent representations for the Dynamic Camera Square task using diffusion policy trained with (left) a standard behavioral cloning objective from scratch and (right) CLASS pre-training. Each colored spline represents an individual episode, with 20 episodes shown per plot. The embeddings are projected using t-SNE with a perplexity of 100.

J.2 Nearest Neighbors

We additionally visualize nearest neighbors using CLASS-trained representations in Figure 14 and compare to BC-trained representations in Figure 15. Specifically, for Square task with varying viewpoints, it is observed that CLASS allows the representation from different viewpoints to be clustered if the future expert action sequences match, whereas BC exhibits little variations in viewpoints between neighbors.

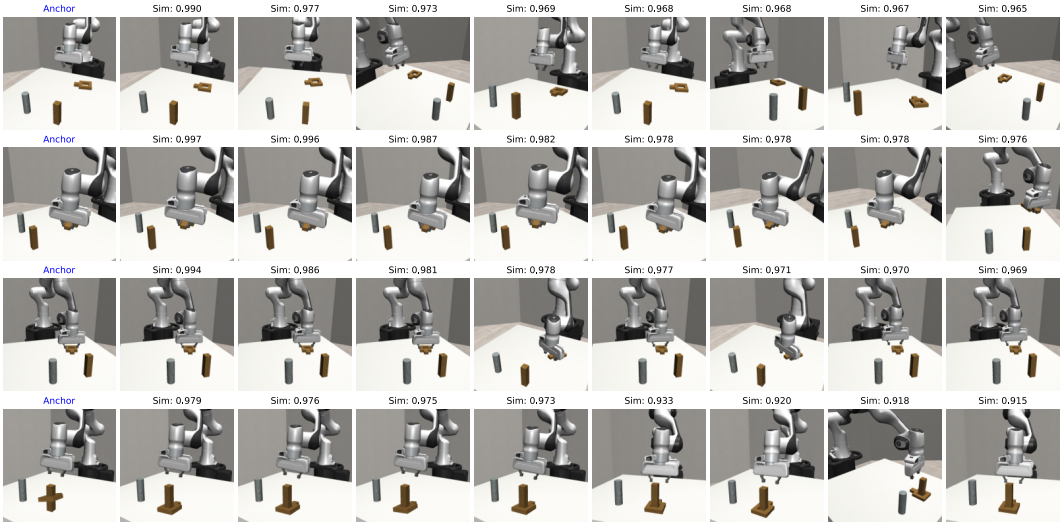


Figure 14: Visualizations of top 8 nearest neighbors for Dynamic Camera Square task with **CLASS** pre-trained policy. The nearest neighbors are retrieved using cosine similarity in the latent space. It can be seen that similar environment scenes with varying viewpoints are identified as closest neighbors.

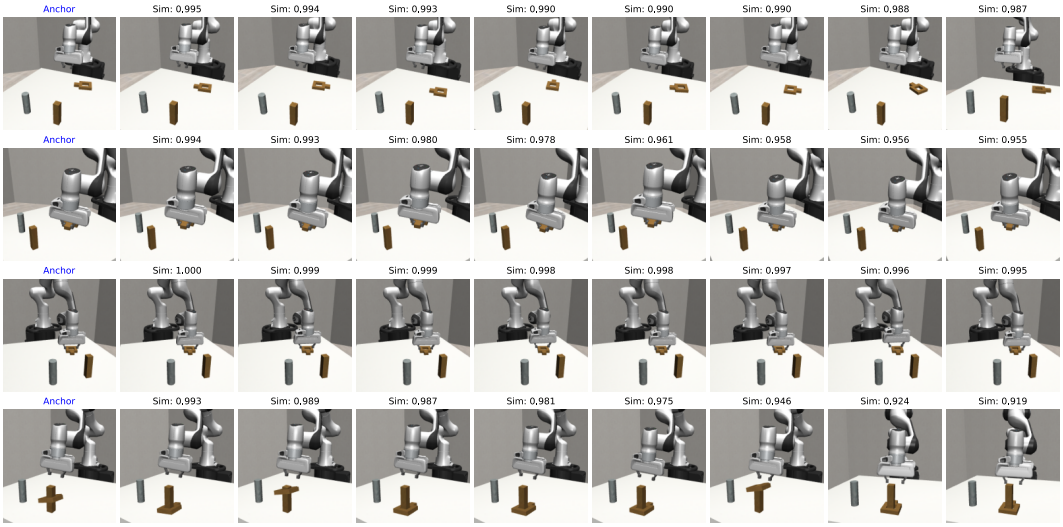


Figure 15: Visualizations of top 8 nearest neighbors for Dynamic Camera Square task with **BC** trained diffusion policy with ImageNet pre-trained weights. Unlike CLASS pre-trained policy, nearest neighbors exhibit little variations in viewpoints.

K Real Robot Experiments

We conduct real-world experiments using a Franka Research 3 robot with the UMI gripper [42]. Visual observations are captured by an Intel RealSense D435 camera mounted on a tripod, recording RGB images at a resolution of 1280×720. These images are subsequently cropped and resized to $256 \times 256 \times 3$. All experiments are executed on a single workstation equipped with an NVIDIA RTX 4090 GPU. Detailed descriptions of initialization, subtasks, and task completion criteria are provided below.

Two-Stack

- **Initialized:** The red and green cubes are randomly placed within a predefined $30\,\text{cm} \times 15\,\text{cm}$ area.
- **Grasped:** The robot successfully picks up the red cube.
- **Stacked:** The red cube is placed stably on top of the green cube.

Mug-Hang

- **Initialized:** A mug is placed at a random pose on the table within a predefined $10\,\text{cm} \times 10\,\text{cm}$ bounding box, with the handle visible from the agent's view.
- **Grasped:** The robot successfully grasps the mug.
- **Hung:** The mug is hung by its handle on a horizontal rod of the mug tree.

Toaster-Load

- **Initialized:** A mock soft bread slice is placed vertically using a support handle. The toaster is placed at a random orientation with a pre-defined position.
- **Grasped:** The robot successfully lifts the bread slice from the holder.
- **Loaded:** The bread is fully inserted into the toaster slot.

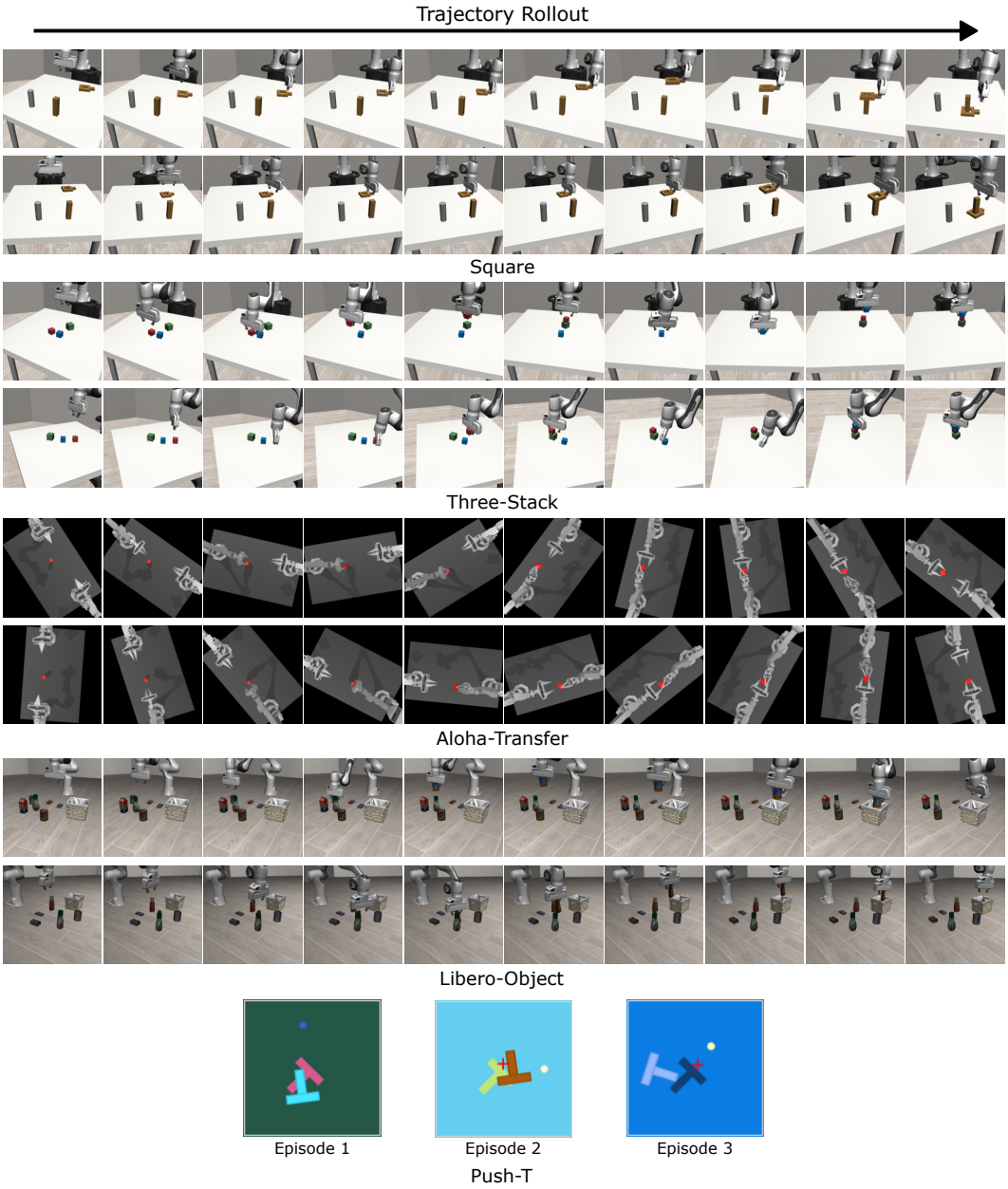


Figure 16: Visualizations of heterogeneous episodes in the simulation tasks. For all tasks except Push-T, trajectories progress from left to right.

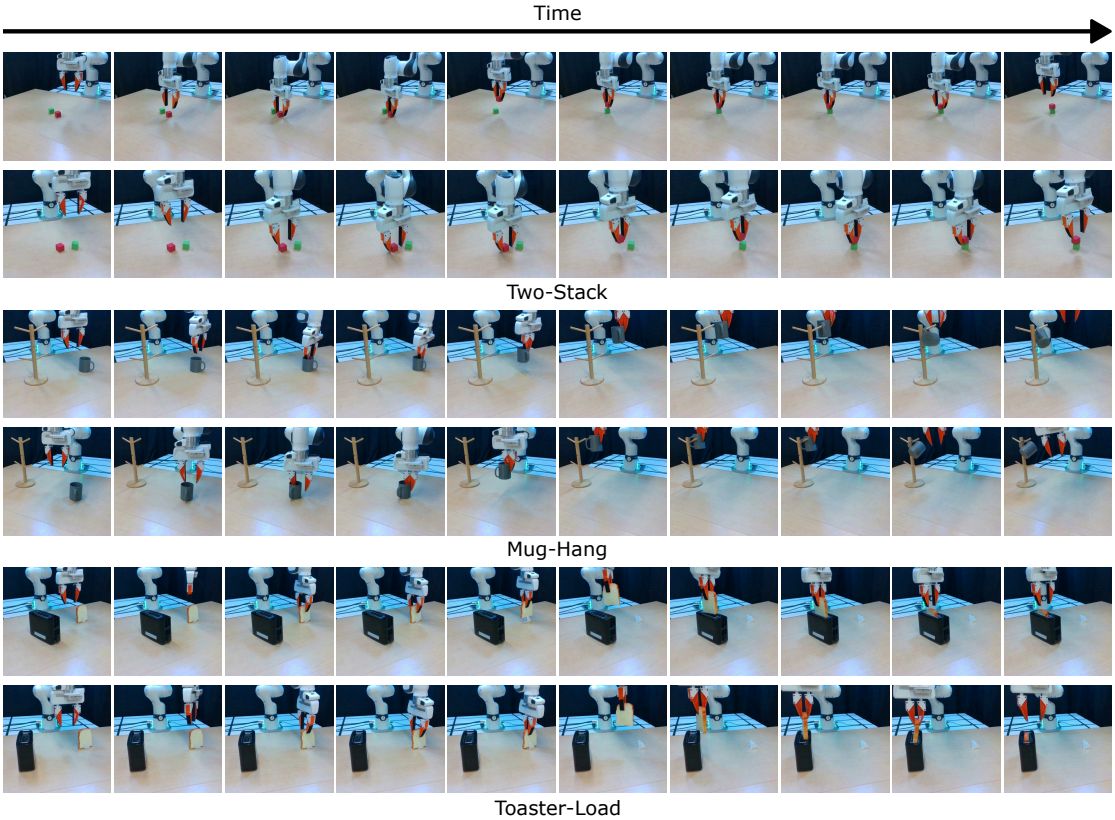


Figure 17: Visualizations of successful evaluation rollouts for real-world tasks. Before each episode, the camera is randomly placed to introduce the visual shifts.