# GOAL-CONDITIONED MULTI-OBJECT MANIPULATION VIA COMPOSITIONAL ACTIONS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Goal-conditioned reinforcement learning (GCRL) for multi-object manipulation from offline images promises robust, scalable, and generalizable control, but existing generative subgoal methods struggle in test and out-of-distribution settings where combinations of object state-to-goal trajectories lie far from the training distribution. We present COMPositional ACTions (COMPACT), a compositional GCRL framework that learns generalizable manipulation policies end-to-end. COMPACT combines an object-centric encoder that discovers objects via feature connectivity and factorizes them into appearance and pose representations, with a compositional policy that aligns state and goal objects by appearance and selects the least displaced object for action prediction. The selected state-goal object pair, together with a learned pose difference embedding, conditions a goal-conditioned policy trained with implicit Q-learning. In multi-object manipulation environments based on OGBench and Push-T, COMPACT improves task success and sample efficiency over state-of-the-art generative subgoal and flat baselines. Importantly, COMPACT is the only method that achieves non-trivial zero-shot performance in settings with an increasing number of objects. Our results demonstrate that object discovery and representation, appearance-based alignment, and compositional action selection provide an effective foundation for data-efficient, generalizable multi-object manipulation from offline visual data.

## 1 INTRODUCTION

Goal-conditioned reinforcement learning (GCRL) for multi-object manipulation on offline image data has the potential to learn robust, scalable, controllable and generalizable object manipulation policies (Kaelbling, 1993; Kostrikov et al., 2022; Park et al., 2025). Recent works typically follow a generative subgoal approach, where latent or image subgoals (see Figure 1a and 1b) are learned from offline trajectories and generated to guide the policy to the goal via closer subgoals (Andrychowicz et al., 2017; Levy et al., 2019; Park et al., 2023; Haramati et al., 2024; Qi et al., 2025; Haramati et al., 2026). These generative approaches aim to address the challenge of high-dimensional, entangled image observations and combinatorial state-goal configurations by decomposing long-horizon tasks into shorter, more tractable subgoal-reaching problems. However, in practice, these methods struggle to produce robust and generalizable policies when evaluated on test and out-of-distribution scenarios, where the required combinations of object-state-to-goal trajectories often lie far outside the training distribution (Haramati et al., 2026). This failure stems from fundamental limitations of generative subgoal approaches for multi-object manipulation: they tend to expend model capacity to irrelevant object features, background details, or controller-specific artifacts and are thus unable to stitch together multiple trajectories to handle unseen combinations of objects (Lin et al., 2023).

To overcome these limitations, we propose a novel compositional GCRL framework which combines a designed object-centric encoder with a new COMPositional ACTions (COMPACT) policy to learn generalizable goal-conditioned policies in an end-to-end manner (see Figure 1c). The object-centric encoder first employs feature-connectivity to discover objects (Foo et al., 2023), then outputs a set of object representations which capture the appearance and pose of each object in the state and goal images. Based on developed theoretical guarantees for generalizable multi-object manipulation, our COMPACT policy then takes in these object representations, performs a state-goal object appearance alignment, and finally selects the top-ranked state-goal object pair to be passed through



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

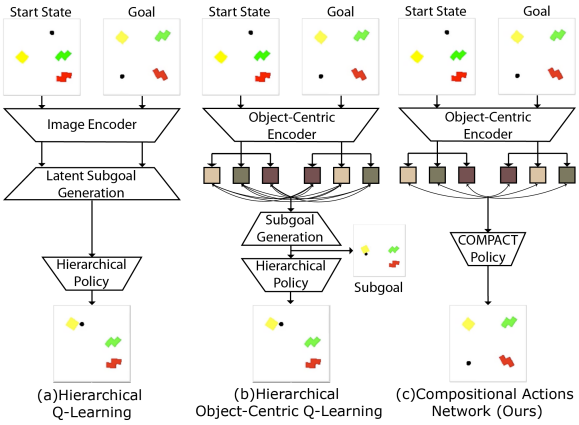



Figure 1: Comparison of existing methods.

the policy head which outputs the next action. The proposed framework is trained end-to-end via Implicit Q-Learning (IQL) objectives (Schaul et al., 2015; Kostrikov et al., 2022).

Empirical evaluation on a suite of multi-object manipulation environments based on (Park et al., 2025; Haramati et al., 2026) show marked improvements in task success of our proposed framework over the state-of-the-art. We also show the superior sample-efficiency of our approach, illustrating its further scalability and practical use in settings where data is limited. Importantly, by increasing the number of objects and testing in a zero-shot manner, we demonstrate that the learned COMPACT policy can robustly achieve its goal in challenging compositional generalization settings, indicating its potential to achieve the long-term objectives of GCRL for multi-object manipulation.

Our contributions are summarized as follows: (1) We propose a novel compositional GCRL framework which obtains object representations and processes them through a compositional actions policy; (2) design a new COMPACT policy grounded on developed theoretical guarantees for linearly bounded learning from offline multi-object manipulation data; (3) demonstrate the superiority of our proposed framework on task success and sample-efficiency in multi-object manipulation environments over the state-of-the-art; and (4) show the generalizability of our approach by presenting improved robustness in zero-shot settings with an increasing number of objects.

## 2 Related Work

Learning robust, generalizable policies for multi-object manipulation has progressed from maximizing past experience in order to address sparse rewards, to imposing structured, object-centric inductive biases on both image-based perception and goal-conditioned control. Since tasks involving the manipulation of multiple objects often requires achieving defined "goal" states, early work established the core idea of goal-conditioned reinforcement learning (GCRL): train a single policy that can reach many goals (Kaelbling, 1993). Hindsight Experience Replay (HER) enabled this in sparse-reward settings by relabeling trajectories so that visited states become achieved goals, allowing learning even when the original goal is not reached (Andrychowicz et al., 2017).

As GCRL scales to longer horizons, conditioning on direct goals becomes inefficient. HAC proposed to address this by learning hierarchical policies that generate intermediate subgoals to guide the agent toward the final goal (Levy et al., 2019). High-level policies propose subgoals while low-level policies reach them. Recent offline methods like HIQL employ action-free value functions and learn a high-level subgoal policy from offline trajectories (Kostrikov et al., 2022; Park et al., 2023).

Most recently, the task of applying GCRL to multi-object image observations has introduced several new challenges involving both image representations and action policies. Holistic image representations prove to be ineffective for learning robust policies, with or without hierarchies (Kostrikov et al., 2022; Park et al., 2023; Haramati et al., 2026). In contrast, learning GCRL policies from object-centric representations of state–goal images (Locatello et al., 2020; Greff et al., 2020; Daniel & Tamar, 2022; Foo et al., 2023; 2025) has shown to be more robust, sample efficient and gener-

alizable (Lin et al., 2023; Ferraro et al., 2025; Zhang et al., 2026). ECRL proposed to compress state–goal images into object-centric representations in order to achieve pixel-based multi-object manipulation under distribution shift (Haramati et al., 2024). EC Diffuser extends this to the offline setting via a generative diffusion-based approach (Qi et al., 2025). Leveraging on subgoals for long-horizon goals, HECRL builds on EC Diffuser and ECRL to propose a hierarchical policy which uses a diffusion model to generate image-based subgoals and trained by IQL (Haramati et al., 2026). Despite the advancements, two drawbacks remain: (1) object-centric representations are pre-trained and fixed a priori instead of being tuned in an end-to-end manner, and (2) subgoal generation fails to generalize learned object-centric manipulation policies under distribution shift.

## 3 Proposed Approach

Our COMPositional ACTions (COMPACT) framework leverages on a compositionality pipeline from input observations to output actions, designed to continuously refine its enconding and policy in an end-to-end, robust and sample efficient manner. COMPACT takes as input a pair of state–goal images and outputs an action representation that determines the next action to be executed in the respective multi-object manipulation environment (see Figure 2).

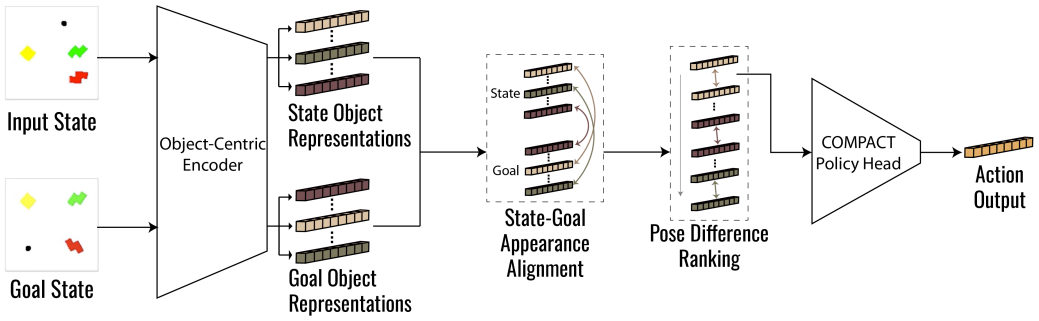



Figure 2: Overview of our COMPACT framework.

The input state image describes the current state of manipulable objects, and the paired goal image depicts the end state that the objects are to end up. COMPACT processes this pair of state–goal images through an object-centric encoder to obtain their respective sets of object representations. The object representations belonging to state and goal are then matched via an appearance alignment algorithm. Once a matching is obtained, the state–goal representation pairs are ranked in order of increasing distance from object state pose to goal pose. Since the object representation pair with the shortest pose distance is computationally the object which requires the shortest action-trajectory to achieve its goal, this pair of shortest-distance representations are passed as input to the COMPACT policy head to output the action representation for the environment. Our COMPACT framework is trained end-to-end via gradient-based implicit Q-learning.

### 3.1 Object–Centric Representations

Our object-centric encoder obtains state and goal sets of object representations via a two-step process of discovery and representation. Let $I^s, I^g \in \mathbb{R}^{H\times W\times 3}$ denote the current state and goal RGB images with dimensions $H \times W$, respectively. The state image describes the current configuration of manipulable objects, while the goal image depicts the configuration that those objects should attain. Each image is individually processed through a constant-initialized pointwise convolution network to obtain an initial set of pixel embeddings $\mathcal{P}^q = \{\mathbf{p}_r^q\}_{r=1}^{HW}, q \in \{s, g\}$. We then group neighboring pixels according to feature-connectivity (Foo et al., 2023). The feature-connectivity distance between two pixels is defined as the shortest-path distance over a spatial-neighborhood graph whose edge weights are determined by distances between neighboring pixel embeddings: $d_{\mathrm{FC}}(r, r') = \mathrm{ShortestPath}\left(\mathbf{p}_r^q, \mathbf{p}_{r'}^q\right)$. Pixels with sufficiently small feature-connectivity distance are assigned to the same object. This produces an unordered set of object masks $\mathcal{M}^q = \{M_k^q\}_{k=1}^{K_q}$, where each mask identifies the pixels assigned to one object.

Once objects have been discovered and identified by their masks, we next define an initial object representation for each object, tuned by gradients. Let $\mathbf{s} \in \mathcal{S}$ denote the full multi-object state, factorized as $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_K)$ with $\mathbf{s}_k \in \mathcal{S}_k$ the state of object $k$. Similarly, let $\mathbf{g} \in \mathcal{G}$ denote the full multi-object goal, factorized as $\mathbf{g} = (\mathbf{g}_1, \ldots, \mathbf{g}_K)$ with $\mathbf{g}_k \in \mathcal{G}_k$ the goal representation of object $k$. We then define the object representations $\mathbf{s}_k = (\mathbf{s}_k^{\text{app}}; \mathbf{s}_k^{\text{pose}}), \mathbf{g}_j = (\mathbf{g}_j^{\text{app}}; \mathbf{g}_j^{\text{pose}})$, where $\mathbf{s}_k$ and $\mathbf{g}_j$ denote state and goal objects, respectively. The appearance and pose components are defined as $\mathbf{s}_k^{\text{app}} = (\mathbf{s}_k^{\text{tex}}, \mathbf{s}_k^{\text{shape}}, \mathbf{s}_k^{\text{scale}}), \mathbf{s}_k^{\text{pose}} = (\mathbf{s}_k^{\text{pos}}, \mathbf{s}_k^{\text{ori}})$, with analogous definitions for the goal objects. The pose-invariant appearance representation is computed from the pixels assigned to each object. For an object mask $M_k$, let $\Omega_k$ denote the set of its constituent pixels and let $\mathbf{p}_r, \mathbf{y}_r$ be the embedding and image-plane coordinates of pixel $r$, respectively. The texture feature of object $k$ is the average of all its constituent pixel embeddings: $\mathbf{s}_k^{\text{tex}} = \frac{1}{|\Omega_k|} \sum_{r \in \Omega_k} \mathbf{p}_r$. To obtain the shape feature, we sample $D$ boundary pixels from the object mask $M_k$ with coordinates $\{\mathbf{y}_{k,1}, \ldots, \mathbf{y}_{k,D}\}$ ordered in clockwise direction. Let $\mathbf{y}_{k,c}$ be the coordinates of the centroid of $M_k$, computed as $\mathbf{y}_{k,c} = \frac{1}{|M_k|} \sum_{r \in M_k} \mathbf{y}_r$. The shape feature $D$ vectors that point from the centroid to each boundary pixel, expressed as a flattened vector $\mathbf{s}_k^{\text{shape}} = \text{vec}([\mathbf{y}_{k,1} - \mathbf{y}_{k,c}, \ldots, \mathbf{y}_{k,D} - \mathbf{y}_{k,c}])$. The scale feature is defined as the length of the longest centroid-to-boundary vector: $\mathbf{s}_k^{\text{scale}} = \max_{j=1,\ldots,D} \|\mathbf{y}_{k,j} - \mathbf{y}_{k,c}\|_2$. The pose representation is computed from coordinates of the pixels assigned to each object. The position component is obtained from the spatial center of the object mask: $\mathbf{s}_k^{\text{pos}} = \mathbf{y}_{k,c}$, The orientation component is obtained from the dominant principal direction of the object mask. Let $\mathbf{\Sigma}_k = \frac{1}{|M_k|} \sum_{r \in M_k} (\mathbf{y}_r - \mathbf{y}_{k,c})(\mathbf{y}_r - \mathbf{y}_{k,c})^\mathsf{T}$. The orientation is the unit eigenvector $\mathbf{o} \in \mathbb{R}^2$ associated with the largest eigenvalue of $\mathbf{\Sigma}_k$: $\mathbf{s}_k^{\text{ori}} = \arg\max_{\|\mathbf{o}\|_2 = 1} \mathbf{o}^\mathsf{T} \mathbf{\Sigma}_k \mathbf{o}$.

### 3.2 Compositional Actions Policy

Our COMPACT policy is motivated by the observation that optimal multi-object goal-conditioned behavior can be decomposed into a sequence of single-object subtasks. We consider an object-oriented Markov decision process (OO-MDP) with $K$ objects per scene drawn from a offline replay dataset of size $N$, with factored state and action spaces and additive per-object rewards (Diuk et al., 2008; Zhao et al., 2022; Jothimurugan et al., 2023). Having obtained the multi-object state and goal representations $\mathbf{s} \in \mathcal{S}$ and $\mathbf{g} \in \mathcal{G}$ as detailed in the previous section, we now formalize the object-oriented MDP. With $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_K)$ where $\mathbf{s}_k \in \mathcal{S}_k$ is the state of object $k$, and $\mathbf{g} = (\mathbf{g}_1, \ldots, \mathbf{g}_K)$ where $\mathbf{g}_k \in \mathcal{G}_k$ is the target state for object $k$. Let $\mathbf{a} \in \mathcal{A}$ denote the action, factorized as $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_K)$ with $\mathbf{a}_k \in \mathcal{A}_k$ the action component for object $k$. We assume that at each time step at most one object is actively controlled, i.e., at most one $\mathbf{a}_k \neq \mathbf{0}$. Under this object-factorized structure, the reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$ decomposes additively over objects: $r(\mathbf{s}, \mathbf{a}, \mathbf{g}) = \sum_{k=1}^{K} r_k(\mathbf{s}_k, \mathbf{a}_k, \mathbf{g}_k)$, where $r_k$ is the per-object reward for object $k$. Let $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denote the transition dynamics, where $\Delta(\mathcal{S})$ is the space of probability distributions over the state space $\mathcal{S}$. The dynamics are object-factorized when at most one object is controlled:

$$p(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) = \prod_{k=1}^{K} p_k(\mathbf{s}'_k \mid \mathbf{s}_k, \mathbf{a}_k) \quad \text{when at most one } \mathbf{a}_k \neq \mathbf{0}, \tag{1}$$

where $p_k : \mathcal{S}_k \times \mathcal{A}_k \to \Delta(\mathcal{S}_k)$ is the per-object transition model for object $k$.

We consider goal-conditioned policies $\pi : \mathcal{S} \times \mathcal{G} \to \Delta(\mathcal{A})$ that map a state–goal pair to a distribution over actions. Let $\gamma \in (0, 1)$ be the discount factor and $H$ the planning horizon (number of time steps per episode). Given a policy $\pi$, we quantify its multi-object generalization performance by the expected discounted return

$$V^\pi(\mathbf{s}, \mathbf{g}) = \mathbb{E}_{\pi,p}\left[ \sum_{t=0}^{H} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{g}) \,\middle|\, \mathbf{s}_0 = \mathbf{s} \right], \tag{2}$$

where the expectation is over trajectories $(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \ldots)$ generated by the policy $\pi$ and the transition dynamics $p$. We analogously define the optimal multi-object value $V^\star(\mathbf{s}, \mathbf{g}) = \max_\pi V^\pi(\mathbf{s}, \mathbf{g})$ under the optimal policy $\pi^\star$. Since the true multi-object dynamics and reward are unknown at training time, $V^\pi$ cannot be directly optimized for all possible object compositions. Theorem 3.1 below shows that we can instead optimize a per-object approximation error that upper-bounds the multi-object suboptimality uniformly over all scenes. Proof of the theorem is provided in Appendix A.

**Theorem 3.1** (Generalization for Compositional Policies). *Let $\pi^{\mathrm{obj}}$ be a goal-conditioned single-object policy whose value function uniformly approximates the optimal single-object value up to error $\varepsilon$:*

$$\sup_{\mathbf{s}_k, \mathbf{g}_k} \left| V_k^{\mathrm{obj}}(\mathbf{s}_k, \mathbf{g}_k) - V_k^{\star}(\mathbf{s}_k, \mathbf{g}_k) \right| \leq \varepsilon, \tag{3}$$

*where $V_k^{\mathrm{obj}}$ and $V_k^{\star}$ are the single-object value functions for object $k$ under $\pi^{\mathrm{obj}}$ and the optimal single-object policy, respectively. Then, any compositional multi-object policy $\pi^{\mathrm{comp}}$ that, at each step, selects one manipulable object $k_t$ and applies $\pi^{\mathrm{obj}}$ to its matched state–goal pair has value $V^{\mathrm{comp}}(\mathbf{s}, \mathbf{g})$ satisfying, for all scenes and goals $\mathbf{g}$,*

$$\left| V^{\mathrm{comp}}(\mathbf{s}, \mathbf{g}) - V^{\star}(\mathbf{s}, \mathbf{g}) \right| \leq K\varepsilon. \tag{4}$$

Intuitively, Theorem 3.1 shows that composing a single-object policy with bounded error $\varepsilon$ over $K$ objects yields linear multi-object suboptimality at most $K\varepsilon$, uniformly over all object compositions. Importantly, since $K$ and the offline replay dataset size $N$ are fixed for a given environment, minimizing $\varepsilon$ directly controls the worst-case multi-object gap for any combination of objects. Following this, we design a compositional actions (COMPACT) policy that (i) selects which object to act on at each step, and (ii) applies a shared single-object head to its state–goal pair (see Appendix B).

Our proposed COMPACT policy aligns objects via appearance, chooses the manipulable object with smallest pose displacement, and fuses its state–goal representation to predict the action. Given the state and goal object representation sets $\mathcal{S}$ and $\mathcal{G}$, we align objects via appearance using previously defined state $\mathbf{s}_k^{\mathrm{app}}$ and corresponding goal $\mathbf{g}_j^{\mathrm{app}}$ appearance features as a combination of texture, shape and scale. For every state object $k$ and goal object $j$, COMPACT computes the appearance similarity $S_{kj} = -\|\mathbf{s}_k^{\mathrm{app}} - \mathbf{g}_j^{\mathrm{app}}\|_2^2$, where objects with similar texture, shape, and scale features receive larger similarity values. Each state object independently selects the goal object with the highest appearance similarity, $\mu(k) = \arg\max_j S_{kj}$. For each matched pair, COMPACT computes a pose distance using both position and orientation. The positional displacement is $\delta_k^{\mathrm{pos}} = \|\mathbf{s}_k^{\mathrm{pos}} - \mathbf{g}_{\mu(k)}^{\mathrm{pos}}\|_2$, while the orientation displacement is $\delta_k^{\mathrm{ori}} = d_{\mathrm{ori}}(\mathbf{s}_k^{\mathrm{ori}}, \mathbf{g}_{\mu(k)}^{\mathrm{ori}})$. These terms are combined as $\delta_k = \lambda \delta_k^{\mathrm{pos}} + (1-\lambda)\delta_k^{\mathrm{ori}}$, where $\lambda$ controls the relative contributions of position and orientation, tuned by gradients. For orientation vectors, we use the sign-invariant distance $d_{\mathrm{ori}}(\mathbf{o}_1, \mathbf{o}_2) = 1 - |\mathbf{o}_1^{\mathsf{T}}\mathbf{o}_2|$ to account for the sign ambiguity of orientations obtained from principal-component analysis. The set of manipulable state objects is then denoted by $\mathcal{I}_{\mathrm{obj}}$. A learnable robot agent representation is identified and tuned but is treated as contextual information and excluded from $\mathcal{I}_{\mathrm{obj}}$.

COMPACT selects the manipulable object with the smallest matched pose distance up to a learned threshold, $k^{\star} = \arg\min_{k \in \mathcal{I}_{\mathrm{obj}}} \delta_k$, where $\mathbf{s}_{k^{\star}}$ is its current state representation and $\mathbf{g}_{\mu(k^{\star})}$ is its matched goal representation. The selected state and goal objects are then defined as $\mathbf{s}^{\star} = \mathbf{s}_{k^{\star}}$ and $\mathbf{g}^{\star} = \mathbf{g}_{\mu(k^{\star})}$. The positional and orientation displacements of the selected object are embedded using a learned pose projection, $\mathbf{r}_{\mathrm{pose}} = \mathrm{MLP}_{\mathrm{pose}}([\delta_{k^{\star}}^{\mathrm{pos}}; \delta_{k^{\star}}^{\mathrm{ori}}])$. The selected state object, its matched goal object, and the pose embedding are fused into the pair representation, $\mathbf{h}_{\mathrm{pair}} = \mathrm{PairFusion}([\mathbf{s}^{\star}; \mathbf{g}^{\star}; \mathbf{r}_{\mathrm{pose}}])$. The COMPACT policy head applies layer normalization and an output multilayer perceptron to the pair representation, producing the action representation $\mathbf{u}_t$. The action representation is then decoded into an environment action, $\mathbf{a}_t = \mathcal{D}_{\psi}(\mathbf{u}_t)$, where $\mathcal{D}_{\psi}(\cdot)$ is the action decoder tuned for each environment.

### 3.3 End-to-End Implicit Q-Learning

COMPACT is trained end-to-end using offline, goal-conditioned implicit Q-learning. The object-centric encoder, appearance-alignment module, pose-difference module, and policy head are jointly optimized from transitions collected in a replay dataset. Let a training transition be $(I_t^s, I^g, \mathbf{a}_t, r_t, I_{t+1}^s)$, where $I_t^s, I_{t+1}^s \in \mathbb{R}^{H \times W \times 3}$ are consecutive state images, $I^g \in \mathbb{R}^{H \times W \times 3}$ is the goal image, $\mathbf{a}_t$ is the executed action, and $r_t$ is the reward. The state and goal images are encoded into object sets $\mathcal{S}_t$ and $\mathcal{G}$, which are aligned and reduced to the selected object pair according to Algorithm 1. We denote the resulting COMPACT representation by $\mathbf{h}_t = \mathbf{h}_{\mathrm{pair}}(\mathcal{S}_t, \mathcal{G})$.

We train COMPACT by minimizing

$$\mathcal{L}_{\mathrm{COMPACT}} = \mathbb{E}_{\mathcal{D}}\left[\mathcal{L}_Q + \lambda_V \mathcal{L}_V + \lambda_{\pi} \mathcal{L}_{\mathrm{policy}}\right], \tag{5}$$

where the expectation is over transitions $(I_t^s, I^g, \mathbf{a}_t, r_t, I_{t+1}^s) \sim \mathcal{D}$.

The critic loss is

$$\mathcal{L}_Q = \left(Q_\omega(\mathbf{h}_t, \mathbf{a}_t) - y_t\right)^2, \quad y_t = r_t + \gamma V_{\bar{\nu}}(\mathbf{h}_{t+1}), \tag{6}$$

where $\gamma$ is the discount factor and $\bar{\nu}$ denotes the parameters of a slowly updated target value network.

The value loss with scalar residual $u$ uses expectile regression with parameter $\tau \in (0.5, 1)$:

$$\mathcal{L}_V = \rho_\tau \left(Q_{\bar{\omega}}(\mathbf{h}_t, \mathbf{a}_t) - V_\nu(\mathbf{h}_t)\right), \quad \rho_\tau(u) = |\tau - \mathbb{I}[u < 0]|u^2, \tag{7}$$

where $\mathbb{I}[\cdot]$ is the indicator function that equals 1 if its argument is true and 0 otherwise.

The policy loss combines advantage-weighted implicit Q-learning and behavior cloning (Kostrikov et al., 2022; Qi et al., 2025). Here $\pi_\phi(\mathbf{a}_t \mid \mathbf{h}_t)$ denotes the policy distribution over actions $\mathbf{a}_t$ conditioned on the COMPACT representation $\mathbf{h}_t$, with parameters $\phi$:

$$\mathcal{L}_{\text{policy}} = (1 - \alpha)\mathcal{L}_{\text{IQL}} + \alpha\mathcal{L}_{\text{BC}}, \qquad \alpha \in [0, 1], \tag{8}$$

where

$$\mathcal{L}_{\text{IQL}} = -w_t \log \pi_\phi(\mathbf{a}_t \mid \mathbf{h}_t), \quad w_t = \exp(\beta A(\mathbf{h}_t, \mathbf{a}_t)), \tag{9}$$

$$\mathcal{L}_{\text{BC}} = -\log \pi_\phi(\mathbf{a}_t \mid \mathbf{h}_t), \quad A(\mathbf{h}_t, \mathbf{a}_t) = Q_{\bar{\omega}}(\mathbf{h}_t, \mathbf{a}_t) - V_\nu(\mathbf{h}_t). \tag{10}$$

Full definitions are provided in Appendix C.

The COMPACT representation $\mathbf{h}_t$ provides structured object-level conditioning for all terms in Equation 5. The appearance-based alignment identifies the state-goal object correspondence, pose-based selection determines the object to be manipulated, and the combined objective trains the policy to imitate dataset actions while emphasizing actions with high estimated advantage.

## 4 Performance Study

We conduct experiments to evaluate the performance of COMPACT in terms of task success performance, sample efficiency and compositional generalization. Adapting from the Push-T (Chi et al., 2025) and the OGBench suite (Park et al., 2025), we use a diverse range of multi-object manipulation environments and tasks to demonstrate its effectiveness across various scenarios:

- High-contact tasks Push-Tetris and Stack-Cube. These tasks involve significant inter-object and agent-object dynamics. Push-Tetris requires fine-grained manipulation of Tetris-like blocks to goal configurations including position and orientation. Stack-Cube contains Pick-and-Place operations which requires stacking cube blocks to the goal configuration.
- Long-horizon tasks PPP-Cube and Scene. These tasks have an extended sequence of subtasks with large permutation. PPP-Cube consists of Pick-Place and Push operations to move cube blocks to the specified goal state. Scene involves manipulation of various object types (button, cube, drawer, window) with varying subtask sequential dependencies.

Each of the four environments involve table-top multi-object manipulation with randomly sampled initial state and final goal states (Haramati et al., 2026). Push-Tetris contains a robotic controller (visually a black dot), while all other environments contain a UR5e robot arm. All perceptual input is in the form of images, with Stack-Cube and PPP-Cube having multiple views for input. A summary of environment characteristics is provided in Table 1.

We compare COMPACT against four state-of-the-art GCRL baselines. HECRL (Haramati et al., 2026) encodes the input into object-centric representations and employs subgoals generated by a Subgoal Diffuser to guide itself towards the goal. ECRL (Haramati et al., 2024) similarly encodes the input into object-centric representations and is trained via implicit Q-learning objectives. HIQL uses hierarchical implicit Q-learning with state-based subgoals (Park et al., 2023). IQL is a flat, goal-conditioned implicit Q-learning baseline with latent state–goal representations (Kostrikov et al., 2022). All methods are trained with 3 objects (excluding Scene) up to 3M state transitions, and we show results of the final checkpoint across 4 seeds. COMPACT's ranking selection is augmented with a learnable weight vector to respect environment-specific ordering (like lowest cube first) for Stack-Cube and Scene. Complete environment and baseline details are provided in Appendix D.

We evaluate all methods using state–goal pixel coverage, task success and Chamfer distance from 100 evaluation episodes. Full definitions and additional details are provided in Appendix E.

Table 1: Summary of environment characteristics.

| Environment | Objects | Object Type | Horizon | Contact | Viewpoints | Transitions |
|---|---|---|---|---|---|---|
| Push-Tetris | 3–7 | Tetris-like | Medium–Long | High | Top | 1M |
| Stack-Cube | 2–4 | Cube | Medium–Long | High | Front and Side | 3M |
| PPP-Cube | 1–6 | Cube | Long | Medium | Front and Side | 2.5M |
| Scene | 3–4 | Varied | Long | Medium | Front | 1.5M |

**Coverage.** Following Haramati et al. (Haramati et al., 2026), we report state-goal pixel coverage for all environments to quantify the state–goal difference in a fine-grained manner. Coverage measures the fraction of goal object pixels that are covered by object pixels in the final state, averaged across all objects. Higher values indicate better pixel-based visual alignment with the goal.

**Task Success.** An episode is successful if all manipulable objects reach their goal positions within a fixed horizon. The success rate is the proportion of successful episodes over $N$ evaluation trials.

**Chamfer Distance.** We report the Chamfer distance between the final state and goal object pixel sets (Haramati et al., 2024). Lower values indicate better alignment with the goal configuration.

### 4.1 Experiments on Multi-Object Manipulation

We first evaluate COMPACT's manipulation performance on the various environments. Table 2 shows the results. As seen from the results for high-contact tasks in Table 2(a), COMPACT significantly outperforms all baselines on Push-Tetris across all metrics, indicating more diverse exploration and more precise multi-object pushing. On Stack-Cube, COMPACT consistently surpasses all other baseline methods significantly, indicating more stable contact-aware policies on the challenging task of fine-grained stacking. Similarly, from the results on long-horizon tasks in Table 2(b), COMPACT surpasses all baselines on PPP-Cube and Scene across all metrics, showing the stability and superiority of the learned COMPACT policy in long-horizon and visually diverse tasks. These results demonstrate that COMPACT scales effectively from medium-horizon, high-contact manipulation to long-horizon scenarios with varied object types.

Figure 3 visualizes sample manipulation trajectories for the respective environments. COMPACT efficiently manipulates the tetris-like blocks in Push-Tetris to their goal position and then orientation. In the manipulation of cubes in Stack-Cube and PPP-Cube, the learned COMPACT policy demonstrates efficient fine-grained manipulation for Pick-Place and Push operations via the UR5e robot arm. Finally, for visually diverse objects in Scene, COMPACT shows efficient and robust manipulation of the varied objects to their goal configuration.

Table 2: Evaluation scores of multi-object manipulation.

(a) High-contact tasks

| | Push-Tetris | | | Stack-Cube | | |
|---|---|---|---|---|---|---|
| Method | Coverage↑ | Success↑ | Chamfer↓ | Coverage↑ | Success↑ | Chamfer↓ |
| IQL | 3.34±0.80 | 0.00±0.00 | 1.13±0.29 | 2.15±0.17 | 0.00±0.00 | 2.74±0.46 |
| HIQL | 4.81±0.98 | 0.00±0.00 | 1.02±0.32 | 2.29±0.23 | 9.97±6.34 | 2.27±0.45 |
| ECRL | 28.85±1.43 | 0.00±0.00 | 0.57±0.17 | 8.97±2.44 | 17.66±1.43 | 1.81±0.38 |
| HECRL | 56.49±2.56 | 50.92±4.60 | 0.20±0.04 | 11.04±2.20 | 36.67±3.21 | 1.76±0.31 |
| COMPACT | **62.12±1.69** | **55.87±4.25** | **0.10±0.02** | **17.57±2.89** | **51.56±5.64** | **1.14±0.43** |

(b) Long-horizon tasks

| | PPP-Cube | | | Scene | | |
|---|---|---|---|---|---|---|
| Method | Coverage↑ | Success↑ | Chamfer↓ | Coverage↑ | Success↑ | Chamfer↓ |
| IQL | 2.16±0.59 | 0.00±0.00 | 1.97±0.13 | 19.74±4.92 | 16.00±4.92 | 0.97±0.27 |
| HIQL | 2.38±0.72 | 2.67±0.58 | 1.93±0.51 | 23.48±3.83 | 17.10±8.22 | 0.71±0.13 |
| ECRL | 12.40±5.36 | 22.07±5.29 | 0.88±0.64 | 53.03±5.86 | 54.00±4.90 | 0.19±0.12 |
| HECRL | 25.17±6.22 | 76.00±5.65 | 0.74±0.74 | 55.15±5.81 | 58.05±4.76 | 0.18±0.11 |
| COMPACT | **32.84±3.19** | **85.33±2.62** | **0.59±0.40** | **63.49±4.84** | **67.30±6.17** | **0.09±0.05** |

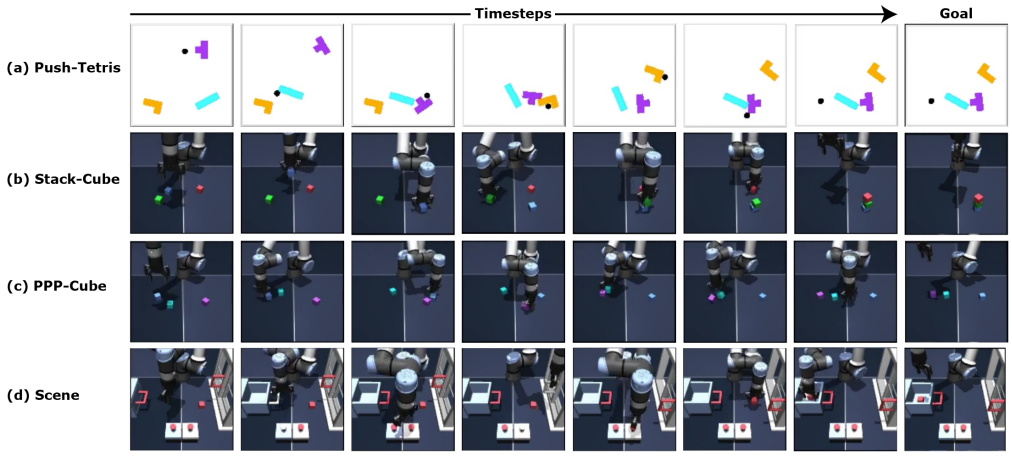
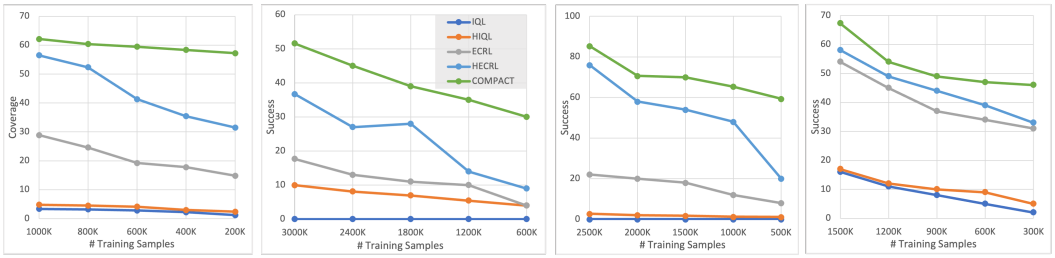



Figure 3: Visualization of sample multi-object manipulation trajectories.

### 4.2 Experiments on Sample Efficiency

One obstacle to robust multi-object manipulation is the availability of a sufficiently large number of offline training samples. Sample efficiency refers to a methods's ability to learn effectively from a relatively small number of samples. Figure 4 shows the performance scores as we decrease the number of training samples in the environments. COMPACT is most robust to decreases in the training samples as compared to all the other methods. The high sample efficiency of COMPACT reduces the need for large, potentially costly or difficult-to-obtain offline data. This makes COMPACT a more practical solution for real-world applications.




(a) Push-Tetris (b) Stack-Cube (c) PPP-Cube (d) Scene

Figure 4: Scores vs decreasing number of training samples.

### 4.3 Experiments on Compositional Generalization

Ideally, a multi-object manipulation framework should be trained to understand common mechanics such as the required positioning of the agent from each object and the physical dynamics between objects, so as to manipulate unseen combinations of objects. In this set of experiments, we compare the generalization ability of COMPACT with the baselines by training the models on 3 objects and testing them on a varying number of objects in a zero-shot manner across the environments.

Table 3 shows the results (coverage for Push-Tetris, success rates otherwise). Increasing the number of objects shows a general decrease in performance for all methods. However, COMPACT experiences the smallest drop in performance and still substantially surpasses the best performing method.

### 4.4 Ablation Studies

Next, we examine the effect of object representations and compositional actions policy on the performance of COMPACT. We implemented two variants of COMPACT: (a) w/o object representations

Table 3: Scores for zero-shot generalizability.

(a) Push-Tetris

| Method | 3-Block | 4-Block | 5-Block | 6-Block | 7-Block |
|---|---|---|---|---|---|
| IQL | 3.34±0.80 | 2.77±0.88 | 2.80±0.99 | 2.92±1.17 | 2.72±1.28 |
| HIQL | 4.81±0.98 | 4.03±1.07 | 3.99±1.99 | 3.81±2.17 | 3.19±2.22 |
| ECRL | 28.85±1.43 | 26.41±1.87 | 19.77±2.10 | 12.38±1.99 | 6.81±2.49 |
| HECRL | 56.49±2.56 | 41.20±2.01 | 31.47±1.50 | 17.39±1.47 | 5.91±1.20 |
| COMPACT | **62.12±1.69** | **60.88±1.97** | **59.17±2.07** | **57.35±1.97** | **56.86±1.70** |

(b) Stack-Cube

| Method | 2-Cube | 3-Cube | 4-Cube |
|---|---|---|---|
| IQL | 1.87±1.29 | 0.00±0.00 | 0.00±0.00 |
| HIQL | 9.97±6.34 | 12.34±5.38 | 0.00±0.00 |
| ECRL | 28.89±12.75 | 17.66±1.435 | 13.52±2.53 |
| HECRL | 56.32±6.90 | 36.67±3.21 | 23.13±3.94 |
| COMPACT | **67.81±5.87** | **51.56±5.64** | **47.81±5.75** |

(c) PPP-Cube

| Method | 3-Cube | 4-Cube | 5-Cube | 6-Cube |
|---|---|---|---|---|
| IQL | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 |
| HIQL | 2.67±0.58 | 1.39±0.89 | 0.00±0.00 | 0.00±0.00 |
| ECRL | 22.07±5.29 | 10.44±4.66 | 3.62±2.13 | 1.26±1.32 |
| HECRL | 76.00±5.65 | 35.73±6.29 | 17.71±6.11 | 9.24±1.31 |
| COMPACT | **85.33±2.62** | **72.90±4.08** | **70.07±3.51** | **64.86±3.46** |

(w/o Obj.). Here, we replace the object-centric encoder with a latent encoder similar to that used in IQL. (b) w/o COMPACT policy (w/o C-Pol.). This network is trained with object representations but without the appearance alignment and pose difference ranking.

Table 4 shows the respective scores for all the environments (coverage for Push-Tetris, success rates otherwise). Without COMPACT policy, there is a substantial drop in performance due to a large increase in search space towards the optimal policy for manipulating multiple objects towards their respective goals. The largest performance drop in all environments is seen when the object representation are replaced with standard latent encodings, indicating the importance of having well-structured object representations for effective multi-object manipulation.

Table 4: Scores for variants of COMPACT.

| Method | Push-Tetris | Stack-Cube | PPP-Cube | Scene |
|---|---|---|---|---|
| COMPACT | **62.12±1.69** | **51.56±5.64** | **85.33±2.62** | **67.30±6.17** |
| w/o C-Pol. | 21.43±1.67 | 27.53±5.92 | 21.93±7.11 | 60.86±6.43 |
| w/o Obj. | 4.05±0.90 | 0.00±0.00 | 0.00±0.00 | 21.93±7.11 |

## 5 Conclusion

We introduced COMPACT, a compositional goal-conditioned reinforcement learning framework for multi-object manipulation from offline images. By discovering objects through feature connectivity, factorizing them into appearance and pose, and aligning state and goal objects via appearance similarity, COMPACT constructs structured object-pair representations that support compositional action selection. The selected object's state-goal representation conditions a goal-conditioned policy trained with implicit Q-learning. Empirical results based on OGBench and Push-T environments show that COMPACT improves task success and sample efficiency over state-of-the-art generative subgoal and flat baselines. Importantly, only COMPACT can generalize robustly to novel object configurations in a zero-shot manner. Our findings highlight the benefits of integrating object representations with compositional goal-conditioned control.

## REFERENCES


Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in Neural Information Processing Systems*, 30, 2017.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Subhabrata Choudhury, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Unsupervised part discovery from contrastive reconstruction. *Advances in Neural Information Processing Systems*, 34:28104–28118, 2021.

Tal Daniel and Aviv Tamar. Unsupervised image representation learning with deep latent particles. *International Conference on Machine Learning (ICML)*, 2022.

Tal Daniel and Aviv Tamar. Ddlp: Unsupervised object-centric video prediction with deep dynamic latent particles. *TMLR*, 2024.

Tal Daniel, Carl Qi, Dan Haramati, Amir Zadeh, Chuan Li, Aviv Tamar, Deepak Pathak, and David Held. Latent particle world models: Self-supervised object-centric stochastic dynamics modeling. *International Conference on Learning Representations (ICLR)*, 2026.

Carlos Diuk, Andre Cohen, and Michael L Littman. An object-oriented representation for efficient reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 240–247, 2008.

Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

Stefano Ferraro, Pietro Mazzaglia, Tim Verbelen, and Bart Dhoedt. Focus: object-centric world models for robotic manipulation. *Frontiers in Neurorobotics*, 19:1585386, 2025.

Alex Foo, Wynne Hsu, and Mong Li Lee. Multi-object representation learning via feature connectivity and object-centric regularization. *Advances in Neural Information Processing Systems*, 36: 60035–60047, 2023.

Alex Foo, Wynne Hsu, and Mong Li Lee. Multi-part object representations via graph structures and co-part discovery. *arXiv preprint arXiv:2512.18192*, 2025.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34:20132–20145, 2021.

Klaus Greff, Sjoerd Van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*, 2020.

Dan Haramati, Tal Daniel, and Aviv Tamar. Entity-centric reinforcement learning for object manipulation from pixels. In *International Conference on Learning Representations (ICLR)*, volume 2024, pp. 29980–30014, 2024.

Dan Haramati, Carl Qi, Tal Daniel, Amy Zhang, Aviv Tamar, and George D Konidaris. Hierarchical entity-centric reinforcement learning with factored subgoal diffusion. In *International Conference on Learning Representations*, volume 2026, pp. 137484–137510, 2026.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and JoÃĢo GM AraÃšjo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

Baoxiong Jia, Yu Liu, and Siyuan Huang. Improving object-centric learning with query optimization. In *International Conference on Learning Representations (ICLR)*, 2023.

Kishor Jothimurugan, Steve Hsu, Osbert Bastani, and Rajeev Alur. Robust subtask learning for compositional generalization. In *International Conference on Machine Learning (ICML)*, pp. 15371–15387. PMLR, 2023.

Leslie Pack Kaelbling. Learning to achieve goals. In *Ijcai*, volume 2, pp. 1094–8, 1993.

Thomas Kipf, Gamaleldin F. Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional Object-Centric Learning from Video. In *International Conference on Learning Representations (ICLR)*, 2022.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *International Conference on Learning Representations (ICLR)*, 2022.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *International Conference on Learning Representations (ICLR)*, 2019.

Weihao Li, Omid Hosseini Jafari, and Carsten Rother. Deep object co-segmentation. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, pp. 638–653. Springer, 2019.

Baihan Lin, Djallel Bouneffouf, and Irina Rish. A survey on compositional generalization in applications. *arXiv preprint arXiv:2302.01067*, 2023.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36:34866–34891, 2023.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, volume 2025, pp. 94937–94982, 2025.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Carl Qi, Dan Haramati, Tal Daniel, Aviv Tamar, and Amy Zhang. Ec-diffuser: Multi-object manipulation via entity-centric behavior generation. In *International Conference on Learning Representations*, volume 2025, pp. 74835–74858, 2025.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning (ICML)*, pp. 1312–1320. PMLR, 2015.

Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, et al. Bridging the gap to real-world object-centric learning. *International Conference on Learning Representations (ICLR)*, 2023.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, pp. 387–395. Pmlr, 2014.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Xudong Wang, Rohit Girdhar, Stella X Yu, and Ishan Misra. Cut and learn for unsupervised object detection and instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3124–3134, 2023a.

Ziyu Wang, Mike Zheng Shou, and Mengmi Zhang. Object-centric learning with cyclic walks between parts and whole. *Advances in Neural Information Processing Systems*, 36:9388–9408, 2023b.

Zizhao Wang, Kaixin Wang, Li Zhao, Peter Stone, and Jiang Bian. Dyn-o: Building structured world models with object-centric representations. *Advances in Neural Information Processing Systems*, 38:121788–121812, 2026.

Weipu Zhang, Adam Jelley, Trevor McInroe, Amos Storkey, and Gang Wang. Object-centric world models from few-shot annotations for sample-efficient reinforcement learning. In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 91366–91394, 2026.

Linfeng Zhao, Lingzhi Kong, Robin Walters, and Lawson LS Wong. Toward compositional generalization in object-oriented world modeling. In *International Conference on Machine Learning (ICML)*, pp. 26841–26864. PMLR, 2022.

# A Upper Bound for Compositional Generalization Error

In this section, we present the proof of Theorem 3.1.

*Proof.* Let the multi-object state and goal be $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_K) \in \mathcal{S}$ and $\mathbf{g} = (\mathbf{g}_1, \ldots, \mathbf{g}_J) \in \mathcal{G}$, with per-object components $\mathbf{s}_k, \mathbf{g}_k \in \mathcal{S}_k$. Let the action be $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_K) \in \mathcal{A}$, with $\mathbf{a}_k \in \mathcal{A}_k$ and at most one $\mathbf{a}_k \neq \mathbf{0}$ per time step.

Recall that the reward function decomposes additively over objects:

$$r(\mathbf{s}, \mathbf{a}, \mathbf{g}) = \sum_{k=1}^{K} r_k(\mathbf{s}_k, \mathbf{a}_k, \mathbf{g}_k), \tag{11}$$

and the transition dynamics decompose across objects when one object is controlled (Equation 1):

$$p(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) = \prod_{k=1}^{K} p_k(\mathbf{s}'_k \mid \mathbf{s}_k, \mathbf{a}_k). \tag{12}$$

Let $\pi^{\mathrm{obj}}$ be a goal-conditioned single-object policy satisfying (Equation 3):

$$\sup_{\mathbf{s}_k, \mathbf{g}_k} \left| V_k^{\mathrm{obj}}(\mathbf{s}_k, \mathbf{g}_k) - V_k^{\star}(\mathbf{s}_k, \mathbf{g}_k) \right| \leq \varepsilon, \tag{13}$$

where $V_k^{\mathrm{obj}}$ and $V_k^{\star}$ are the single-object value functions for object $k$ under $\pi^{\mathrm{obj}}$ and the optimal single-object policy, respectively.

Consider a compositional multi-object policy $\pi^{\mathrm{comp}}$ that, at each time step $t$, selects one manipulable object $k_t$ and applies $\pi^{\mathrm{obj}}$ to its matched state–goal pair. Let $H$ denote the planning horizon (number of time steps per episode) and $\gamma \in (0, 1)$ the discount factor. The multi-object value of $\pi^{\mathrm{comp}}$ is

$$V^{\mathrm{comp}}(\mathbf{s}, \mathbf{g}) = \mathbb{E}_{\pi^{\mathrm{comp}}, p}\left[ \sum_{t=0}^{H} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{g}) \,\middle|\, \mathbf{s}_0 = \mathbf{s} \right]. \tag{14}$$

Substituting the additive reward decomposition (Equation 11) into Equation 14, we obtain

$$\begin{aligned} V^{\mathrm{comp}}(\mathbf{s}, \mathbf{g}) &= \mathbb{E}_{\pi^{\mathrm{comp}}, p}\left[ \sum_{t=0}^{H} \gamma^t \sum_{k=1}^{K} r_k(\mathbf{s}_{k,t}, \mathbf{a}_{k,t}, \mathbf{g}_k) \,\middle|\, \mathbf{s}_0 = \mathbf{s} \right] \\ &= \sum_{k=1}^{K} \mathbb{E}_{\pi^{\mathrm{comp}}, p}\left[ \sum_{t=0}^{H} \gamma^t r_k(\mathbf{s}_{k,t}, \mathbf{a}_{k,t}, \mathbf{g}_k) \,\middle|\, \mathbf{s}_{k,0} = \mathbf{s}_k \right]. \end{aligned} \tag{15}$$

By the object-factorized dynamics (Equation 12) and the assumption that at most one object is controlled at each step, the inner expectation for each object $k$ reduces to the single-object value under $\pi^{\mathrm{obj}}$, up to discounting effects due to the scheduling of objects:

$$V^{\mathrm{comp}}(\mathbf{s}, \mathbf{g}) = \sum_{k=1}^{K} \alpha_k(\mathbf{s}, \mathbf{g})\, V_k^{\mathrm{obj}}(\mathbf{s}_k, \mathbf{g}_k), \tag{16}$$

where $\alpha_k(\mathbf{s}, \mathbf{g}) \in (0, 1]$ are discounting coefficients that depend on the order in which objects are manipulated and satisfy $\sum_{k=1}^{K} \alpha_k(\mathbf{s}, \mathbf{g}) \leq K$. An analogous decomposition holds for the optimal multi-object value:

$$V^{\star}(\mathbf{s}, \mathbf{g}) = \sum_{k=1}^{K} \alpha_k(\mathbf{s}, \mathbf{g})\, V_k^{\star}(\mathbf{s}_k, \mathbf{g}_k). \tag{17}$$

Subtracting Equation 17 from Equation 16 and applying the triangle inequality, we have

$$\begin{aligned} \left| V^{\mathrm{comp}}(\mathbf{s}, \mathbf{g}) - V^{\star}(\mathbf{s}, \mathbf{g}) \right| &= \left| \sum_{k=1}^{K} \alpha_k(\mathbf{s}, \mathbf{g}) \big( V_k^{\mathrm{obj}}(\mathbf{s}_k, \mathbf{g}_k) - V_k^{\star}(\mathbf{s}_k, \mathbf{g}_k) \big) \right| \\ &\leq \sum_{k=1}^{K} \alpha_k(\mathbf{s}, \mathbf{g}) \left| V_k^{\mathrm{obj}}(\mathbf{s}_k, \mathbf{g}_k) - V_k^{\star}(\mathbf{s}_k, \mathbf{g}_k) \right|. \end{aligned} \tag{18}$$

Using the uniform per-object error bound (Equation 13), we obtain

$$\begin{aligned}\left|V^{\mathrm{comp}}(\mathbf{s},\mathbf{g}) - V^{\star}(\mathbf{s},\mathbf{g})\right| &\le \sum_{k=1}^{K} \alpha_k(\mathbf{s},\mathbf{g})\,\varepsilon \\ &\le K\varepsilon.\end{aligned} \tag{19}$$

This establishes the bound in Theorem 3.1:

$$\left|V^{\mathrm{comp}}(\mathbf{s},\mathbf{g}) - V^{\star}(\mathbf{s},\mathbf{g})\right| \le K\varepsilon. \tag{20}$$

□

Elaborating on the main text, Equation 20 shows that the multi-object suboptimality is bounded by $K\varepsilon$, where $\varepsilon$ is the worst-case single-object approximation error. Since $K$ (objects per scene) and the offline replay dataset size $N$ are fixed for a given environment, minimizing $\varepsilon$ directly controls the worst-case multi-object gap. This motivates the COMPACT policy design: (i) select which object to act on at each step via appearance alignment and pose-based selection, and (ii) apply a shared single-object action head to the selected state–goal pair, thereby minimizing $\varepsilon$ through end-to-end training on the offline replay dataset.

## B Compositional Actions Policy

**Algorithm 1** COMPACT policy. The input is a pair of state and goal representation sets, each containing objects of dimension $D_{\mathrm{obj}}$. The policy outputs a single action $\hat{\mathbf{a}}_t$ by (i) aligning state and goal objects using appearance features, (ii) selecting the manipulable object with the smallest pose displacement, and (iii) fusing the selected state-goal representation pair to predict the action.

1: **Input:** State objects $\mathcal{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_K\}$ and goal objects $\mathcal{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_J\}$
2: **Output:** Action $\hat{\mathbf{a}}_t$
3: **for** $k = 1$ to $K$ **do**
4: $\quad \mathbf{s}_k^{\mathrm{app}} \leftarrow \left(\mathbf{s}_k^{\mathrm{tex}}; \mathbf{s}_k^{\mathrm{shape}}; \mathbf{s}_k^{\mathrm{scale}}\right)$
5: $\quad$ **for** $j = 1$ to $K$ **do**
6: $\quad\quad \mathbf{g}_j^{\mathrm{app}} \leftarrow \left(\mathbf{g}_j^{\mathrm{tex}}; \mathbf{g}_j^{\mathrm{shape}}; \mathbf{g}_j^{\mathrm{scale}}\right)$
7: $\quad\quad S_{kj} \leftarrow -\|\mathbf{s}_k^{\mathrm{app}} - \mathbf{g}_j^{\mathrm{app}}\|_2^2$
8: $\quad$ **end for**
9: $\quad \mu(k) \leftarrow \arg\max_j S_{kj}$
10: $\quad \delta_k \leftarrow \lambda \|\mathbf{s}_k^{\mathrm{pos}} - \mathbf{g}_{\mu(k)}^{\mathrm{pos}}\|_2 + (1-\lambda)\, d_{\mathrm{ori}}\left(\mathbf{s}_k^{\mathrm{ori}}, \mathbf{g}_{\mu(k)}^{\mathrm{ori}}\right)$
11: **end for**
12: $\mathcal{I}_{\mathrm{obj}} \leftarrow \{k \mid \mathbf{s}_k \text{ is a manipulable object}\}$
13: $k^{\star} \leftarrow \arg\min_{k \in \mathcal{I}_{\mathrm{obj}}} \delta_k$
14: $\mathbf{s}^{\star} \leftarrow \mathbf{s}_{k^{\star}}$; $\mathbf{g}^{\star} \leftarrow \mathbf{g}_{\mu(k^{\star})}$
15: $\mathbf{r}_{\mathrm{pose}} \leftarrow \mathrm{MLP}_{\mathrm{pose}}\left([\delta_{k^{\star}}^{\mathrm{pos}}; \delta_{k^{\star}}^{\mathrm{ori}}]\right)$
16: $\mathbf{h}_{\mathrm{pair}} \leftarrow \mathrm{PairFusion}\left([\mathbf{s}^{\star}; \mathbf{g}^{\star}; \mathbf{r}_{\mathrm{pose}}]\right)$
17: $\mathbf{u}_t \leftarrow \mathrm{MLP}_{\mathrm{out}}\left(\mathrm{LayerNorm}(\mathbf{h}_{\mathrm{pair}})\right)$
18: $\mathbf{a}_t \leftarrow \mathcal{D}_{\psi}(\mathbf{u}_t)$
19: **return** $\mathbf{a}_t$

Our proposed COMPositional ACTions (COMPACT) policy aligns objects via appearance, chooses the manipulable object with smallest pose displacement, and fuses its state–goal representation to predict the action. The algorithm is detailed above. Given the state and goal object representation sets $\mathcal{S}$ and $\mathcal{G}$, COMPACT first constructs appearance features for each object by concatenating its texture, shape, and scale representations. Specifically, the state appearance feature is $\mathbf{s}_k^{\mathrm{app}} = (\mathbf{s}_k^{\mathrm{tex}}; \mathbf{s}_k^{\mathrm{shape}}; \mathbf{s}_k^{\mathrm{scale}})$, while the corresponding goal appearance feature is $\mathbf{g}_j^{\mathrm{app}} = (\mathbf{g}_j^{\mathrm{tex}}; \mathbf{g}_j^{\mathrm{shape}}; \mathbf{g}_j^{\mathrm{scale}})$ (Lines 3–6 of Algorithm 1). For every state object $k$ and goal object $j$, COMPACT computes the appearance similarity $S_{kj} = -\|\mathbf{s}_k^{\mathrm{app}} - \mathbf{g}_j^{\mathrm{app}}\|_2^2$ (Line 7). Thus, objects with

similar texture, shape, and scale features receive larger similarity values. Each state object independently selects the goal object with the highest appearance similarity, $\mu(k) = \arg\max_j S_{kj}$ (Line 9). For each matched pair, COMPACT computes a pose distance using both position and orientation. The positional displacement is $\delta_k^{\text{pos}} = \|\mathbf{s}_k^{\text{pos}} - \mathbf{g}_{\mu(k)}^{\text{pos}}\|_2$, while the orientation displacement is $\delta_k^{\text{ori}} = d_{\text{ori}}(\mathbf{s}_k^{\text{ori}}, \mathbf{g}_{\mu(k)}^{\text{ori}})$. These terms are combined as $\delta_k = \lambda\,\delta_k^{\text{pos}} + (1-\lambda)\,\delta_k^{\text{ori}}$ (Line 10), where $\lambda$ controls the relative contributions of position and orientation. For orientation vectors, we use the sign-invariant distance $d_{\text{ori}}(\mathbf{o}_1, \mathbf{o}_2) = 1 - |\mathbf{o}_1^{\mathsf{T}}\mathbf{o}_2|$ to account for the sign ambiguity of orientations obtained from principal-component analysis. The set of manipulable state objects is then denoted by $\mathcal{I}_{\text{obj}}$ (Line 12). The robot arm may be represented in the state and goal object sets, but is treated as contextual information and excluded from $\mathcal{I}_{\text{obj}}$.

COMPACT selects the manipulable object with the smallest matched pose distance up to a learned threshold, $k^\star = \arg\min_{k \in \mathcal{I}_{\text{obj}}} \delta_k$ (Line 13), where $\mathbf{s}_{k^\star}$ is its current state representation and $\mathbf{g}_{\mu(k^\star)}$ is its matched goal representation. The selected state and goal objects are then defined as $\mathbf{s}^\star = \mathbf{s}_{k^\star}$ and $\mathbf{g}^\star = \mathbf{g}_{\mu(k^\star)}$ (Line 14). The positional and orientation displacements of the selected object are embedded using a learned pose projection, $\mathbf{r}_{\text{pose}} = \text{MLP}_{\text{pose}}([\delta_{k^\star}^{\text{pos}}; \delta_{k^\star}^{\text{ori}}])$ (Line 15). The selected state object, its matched goal object, and the pose embedding are fused into the pair representation, $\mathbf{h}_{\text{pair}} = \text{PairFusion}([\mathbf{s}^\star; \mathbf{g}^\star; \mathbf{r}_{\text{pose}}])$ (Line 16). The COMPACT policy head applies layer normalization and an output multilayer perceptron to the pair representation, producing the action representation $\mathbf{u}_t$ (Line 17). The action representation is then decoded into an environment action, $\mathbf{a}_t = \mathcal{D}_\psi(\mathbf{u}_t)$, where $\mathcal{D}_\psi(\cdot)$ is the action decoder for the respective environment (Line 18).

## C Training Objectives

Let a training transition be $(I_t^s, I^g, \mathbf{a}_t, r_t, I_{t+1}^s)$, where $I_t^s, I_{t+1}^s \in \mathbb{R}^{H\times W\times 3}$ are consecutive state images, $I^g \in \mathbb{R}^{H\times W\times 3}$ is the goal image, $\mathbf{a}_t$ is the executed action, and $r_t$ is the reward. The state and goal images are encoded into object sets $\mathcal{S}_t = \{\mathbf{s}_{t,1}, \ldots, \mathbf{s}_{t,K}\}$ and $\mathcal{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_J\}$, which are aligned and reduced to the selected object pair according to Algorithm 1. We denote the resulting COMPACT representation by $\mathbf{h}_t = \mathbf{h}_{\text{pair}}(\mathcal{S}_t, \mathcal{G})$.

The goal-conditioned action-value function and state-value function are parameterized as $Q_\omega(\mathbf{h}_t, \mathbf{a}_t)$ and $V_\nu(\mathbf{h}_t)$, respectively. The target value for a transition is

$$y_t = r_t + \gamma V_{\bar{\nu}}(\mathbf{h}_{t+1}), \tag{21}$$

where $\gamma$ is the discount factor and $\bar{\nu}$ denotes the parameters of a slowly updated target value network.

The critic is trained using the Bellman regression objective

$$\mathcal{L}_Q(\omega) = \mathbb{E}_{(I_t^s, I^g, \mathbf{a}_t, r_t, I_{t+1}^s)\sim\mathcal{D}}\left[\left(Q_\omega(\mathbf{h}_t, \mathbf{a}_t) - y_t\right)^2\right]. \tag{22}$$

The value function is estimated using expectile regression. For a scalar residual $u$, define $\rho_\tau(u) = |\tau - \mathbb{I}[u<0]|u^2$, where $\tau \in (0.5, 1)$ is the expectile parameter and $\mathbb{I}[\cdot]$ is the indicator function that equals 1 if its argument is true and 0 otherwise. The value-function objective is

$$\mathcal{L}_V(\nu) = \mathbb{E}_{(I_t^s, I^g, \mathbf{a}_t)\sim\mathcal{D}}\left[\rho_\tau\left(Q_{\bar{\omega}}(\mathbf{h}_t, \mathbf{a}_t) - V_\nu(\mathbf{h}_t)\right)\right], \tag{23}$$

where $\bar{\omega}$ denotes the parameters of a target critic.

The advantage estimate is

$$A(\mathbf{h}_t, \mathbf{a}_t) = Q_{\bar{\omega}}(\mathbf{h}_t, \mathbf{a}_t) - V_\nu(\mathbf{h}_t). \tag{24}$$

The advantage-weighted implicit policy objective is

$$\mathcal{L}_{\text{IQL}}(\phi) = -\mathbb{E}_{(I_t^s, I^g, \mathbf{a}_t)\sim\mathcal{D}}\left[w_t \log \pi_\phi\left(\mathbf{a}_t \mid \mathbf{h}_t\right)\right], \tag{25}$$

where

$$w_t = \exp\left(\beta A(\mathbf{h}_t, \mathbf{a}_t)\right). \tag{26}$$

The behavior-cloning loss is

$$\mathcal{L}_{\text{BC}}(\phi) = -\mathbb{E}_{(I_t^s, I^g, \mathbf{a}_t)\sim\mathcal{D}}\left[\log \pi_\phi\left(\mathbf{a}_t \mid \mathbf{h}_t\right)\right]. \tag{27}$$

The final policy objective combines advantage-weighted implicit policy learning and behavior cloning:

$$\mathcal{L}_{\text{policy}}(\phi) = (1-\alpha)\mathcal{L}_{\text{IQL}}(\phi) + \alpha\mathcal{L}_{\text{BC}}(\phi), \qquad \alpha \in [0,1]. \tag{28}$$

The total COMPACT training objective is

$$\mathcal{L}_{\text{COMPACT}} = \mathbb{E}_{(I_t^s, I^g, \mathbf{a}_t, r_t, I_{t+1}^s)\sim\mathcal{D}}\Big[\mathcal{L}_Q + \lambda_V\mathcal{L}_V + \lambda_\pi\mathcal{L}_{\text{policy}}\Big], \tag{29}$$

where $\lambda_V$ and $\lambda_\pi$ control the contributions of the value and policy objectives. The parameters are updated by gradient descent:

$$(\theta, \omega, \nu, \phi) \leftarrow (\theta, \omega, \nu, \phi) - \eta\nabla\mathcal{L}_{\text{COMPACT}}, \tag{30}$$

where $\theta$ denotes the parameters of the object-centric encoder and $\eta$ is the learning rate.

# D Environments and Baselines

## D.1 Environments

We evaluate on four environments based on Push-T (Chi et al., 2025) and OGBench suite (Park et al., 2025).

**Push-Tetris.** This environment tests multi-object manipulation under dense contact and collision constraints. Adapted from the Push-T environment (Chi et al., 2025) to a multi-object manipulation task using seven distinct Tetris-like blocks, each with a fixed color, where the agent must push them to goal configurations specifying both position and orientation in 2D space using 128×128 RGB pixel observations and a 2D action space of agent position deltas. Success is defined as achieving over 70% pixel coverage between the final and goal states. The dataset is collected via a random policy constrained to a fixed radius around an object, yielding highly suboptimal trajectories, unlike the original Push-T's expert demonstrations. Evaluation is performed on agents trained for 1M gradient steps, tested across 100 randomly sampled initial and goal configurations with a maximum episode length of 1000 timesteps for 3-object scenarios. Key differences from the original task include: (1) replacing the single T-block with seven Tetris-like shapes, (2) using semi-random rather than expert data, and (3) specifying goals via a separate image instead of a shaded region overlaid on the observation.

**Stack-Cube.** The Stack-Cube environment extends OGBench Cube to a multi-object, multi-color 3D manipulation task where an agent must move cubes between random initial and goal positions using two 128×128 RGB views (front and side) or factored state observations encoding each entity's 3D position and yaw (Haramati et al., 2026). A dataset of 3M transitions (3,000 episodes × 1,000 steps) involving three cubes is used, collected by a noisy scripted policy that performs Pick-and-Place operations with a high probability of stacking (Haramati et al., 2026). Crucially, the scripted policy includes a recovery mechanism: if a lower block in a partial stack is accidentally displaced during stacking, the policy first restores those blocks before re-attempting to place the target cube on top, making the data significantly more goal-directed and enabling robust RL learning. Evaluation is conducted on agents trained for 3M gradient steps, tested across 200 randomly sampled initial and goal configurations with a maximum episode length of 1,000 timesteps for the 3-cube setting. Key differences from OGBench include: (1) multi-view 128×128 pixels vs. single-view 64×64; (2) removal of OGBench's semi-transparent arm and purple end-effector highlight; (3) per-episode random coloring of cubes from six options rather than fixed colors tied to cube count; (4) inclusion of push operations in data collection; and (5) shorter dataset episodes (400 vs. 1,000 steps).

**PPP-Cube.** The PPP-Cube environment extends OGBench Cube to a multi-object, multi-color 3D manipulation task where an agent must move cubes between random initial and goal positions using two 128×128 RGB views (front and side) or factored state observations encoding each entity's 3D position and yaw (Haramati et al., 2026). Each episode randomly assigns distinct colors to the cubes from a fixed palette of six, increasing state-space complexity and enabling tests of generalization to more objects and longer horizons without introducing new colors. The 5D action space comprises deltas in gripper 3D position, yaw, and opening, and a goal is achieved when all cubes lie within a threshold distance of their target positions. The dataset contains 3M transitions (7,500 episodes × 400 steps) of diverse interactions—pick, place, and push—generated by a noisy scripted policy that

either picks and places a randomly chosen cube or pushes it from a random face. Evaluation uses agents trained for 2.5M gradient steps, tested on 100 random initial–goal pairs with a maximum episode length of 1,000 timesteps for 3 cubes. Key differences from OGBench include: (1) multi-view 128×128 pixels vs. single-view 64×64; (2) removal of OGBench's semi-transparent arm and purple end-effector highlight; (3) per-episode random coloring of cubes from six options rather than fixed colors tied to cube count; (4) inclusion of push operations in data collection; and (5) shorter dataset episodes (400 vs. 1,000 steps).

**Scene.** The Scene environment adopts OGBench's Visual Scene setup, requiring manipulation of diverse objects (button, cube, drawer, window) in 3D space using single-view 128×128 RGB images and a 5D action space (gripper 3D position deltas, yaw, and opening), with success defined as all objects lying within a threshold distance of their goal positions (Haramati et al., 2026). To resolve goal ambiguities introduced by their more realistic visual rendering—where the robot arm is opaque and can occlude objects—tasks 4 and 5 are modified: instead of specifying a goal with the cube inside a closed (and thus visually hidden) drawer, the goal is redefined as placing the cube in the drawer only when it is opened and locked. The dataset mirrors OGBench's collection policy, using a noisy scripted agent that executes applicable subtasks (e.g., inserting the cube into the drawer only when open) in random order, yielding 1M transitions across 1,000 episodes of 1,000 steps each. Evaluation is performed on agents trained for 1.5M gradient steps, tested on 50 randomly perturbed initial–goal configurations per task with a maximum episode length of 1,000 timesteps. Differences from the original include the visual and task modifications noted in PPP-Cube items (1)–(2), plus the targeted adjustments to tasks 4 and 5 to ensure unambiguous goal specification under occlusion.

### D.2 Baselines

**HECRL** (Haramati et al., 2026) uses hierarchical entity-centric reinforcement learning with factored subgoal diffusion. It generates entity-factored subgoals and trains a low-level goal-conditioned policy using implicit Q-learning. We implement this baseline on top of an ECRL (Haramati et al., 2024) agent trained with IQL (see EC-IQL below for details). For the Subgoal Diffuser, EC-Diffuser (Qi et al., 2025) is adapted to generate subgoals conditioned on object-centric states and goals. Specifically, action inputs are removed entirely and condition on initial state and goal entities as clean inputs that are not denoised. Learned additive embeddings are used to encode each entity's affiliation to one of three sets (state, goal, or noisy subgoal) and, in the multi-view setting, to one of two views. The Diffuser architecture is an 8-layer Transformer conditioned on the diffusion timestep via adaptive layer normalization (AdaLN). We base our implementation on the official HECRL repository[1].

**ECRL** (Haramati et al., 2024) is implemented as an agent adapted to the offline learning setting by integrating the Entity Interaction Transformer (EIT) architecture with IQL. The EIT architecture for the Q-function is modified by replacing the action entity conditioning with AdaLN conditioning. We base our implementation on the official ECRL repository[2].

**HIQL** (Park et al., 2023) uses hierarchical implicit Q-learning with state-based subgoals for goal-conditioned tasks. It is trained with an underlying IQL agent and extract the low-level policy using a combination of Deep Deterministic Policy Gradient (DDPG) (Silver et al., 2014) and behavioural cloning (BC) (Pomerleau, 1991; Fujimoto & Gu, 2021). Unlike the other IQL-based methods in this work, and consistent with the original HIQL, we train the low-level policy on the subgoal distribution rather than the goal distribution. The high-level policy is trained with AWR using the IQL value function. At test time, the subgoal is held fixed for the same duration as the Diffuser subgoals. All agent components are 4-layer MLPs with a hidden dimension of 512. We base our implementation on the official OGBench repository[3].

**IQL** (Kostrikov et al., 2022) is a flat, goal-conditioned implicit Q-learning baseline without object-centric representations or hierarchical decomposition. All agent components are 4-layer MLPs with a hidden dimension of 512. We base our implementation on the official OGBench repository[4].

[1] https://github.com/DanHrmti/HECRL
[2] https://github.com/DanHrmti/ECRL
[3] https://github.com/seohongpark/ogbench
[4] https://github.com/seohongpark/ogbench

All methods are implemented in PyTorch, building on official codebases and LeanRL [5], an efficient variant of CleanRL (Huang et al., 2022). HECRL and ECRL are trained from latent image-based representations extracted using a pretrained DLPv2 (Daniel & Tamar, 2024). Standard agents HIQL and IQL are trained from latent image-based representations extracted using a pretrained VQ-VAE (Van Den Oord et al., 2017). Low-level policies are distilled from IQL Q-functions using DDPG and BC (Fujimoto & Gu, 2021). For value function training, goals are sampled uniformly from future states in the same trajectory with probability 0.8, and otherwise set to the immediate next state. Low-level policy training goals are likewise sampled uniformly from future states in the same trajectory (except in HIQL). COMPACT, HECRL and ECRL use the same object-centric EIT architectures for the value function and Q-function networks, while HIQL and IQL use MLP based networks for the latent representations. Shared hyperparameters are reported in Table 5.

Table 5: Shared hyperparameters.

| Hyperparameter | Value |
|---|---|
| Batch size | 512 |
| Learning rate | $3 \times 10^{-4}$ |
| Gradient clip norm | 20 |
| Discount factor $\gamma$ | 0.99 |
| Target smoothing coefficient $\tau$ | 0.005 |
| IQL/HIQL expectile | 0.9 |
| AWR temperature $\beta$ | 3.0 |
| Subgoal Diffuser diffusion steps | 10 |
| Subgoal $K$ | 50 |
| EIT attention dimension | 64 |
| EIT attention heads | 8 |
| EIT hidden dimension | 256 |
| MLP layers | 4 |
| MLP hidden dimension | 512 |

## E Evaluation Metrics

Let $\mathcal{E}$ denote a test environment and $\pi$ a policy. An evaluation episode $e$ consists of a horizon $T_e$, a sequence of state images $\{I_t^{s,(e)}\}_{t=0}^{T_e}$, and a fixed goal image $I^{g,(e)}$.

**Coverage.** Following Haramati et al. (Haramati et al., 2026), we report state-goal pixel coverage for environments that require fine-grained manipulation. For episode $e$, let $M_{\text{state}}^{(e)}$ and $M_{\text{goal}}^{(e)}$ denote the binary masks of manipulable object pixels in the final state image $I_{T_e}^{s,(e)}$ and goal image $I^{g,(e)}$, respectively. The coverage is

$$\text{Cov}^{(e)}(\pi) = \frac{\left|M_{\text{state}}^{(e)} \cap M_{\text{goal}}^{(e)}\right|}{\left|M_{\text{goal}}^{(e)}\right|}. \tag{31}$$

The average coverage over $N$ evaluation episodes is

$$\overline{\text{Cov}}(\pi; \mathcal{E}) = \frac{1}{N} \sum_{e=1}^{N} \text{Cov}^{(e)}(\pi). \tag{32}$$

**Chamfer Distance.** We report the Chamfer distance between the final state and goal configurations. Let $\mathcal{O}_{\text{state}}^{(e)}$ and $\mathcal{O}_{\text{goal}}^{(e)}$ denote the sets of object pixel coordinates in $M_{\text{state}}^{(e)}$ and $M_{\text{goal}}^{(e)}$, respectively. The Chamfer distance is

$$\text{CD}^{(e)}(\pi) = \frac{1}{|\mathcal{O}_{\text{state}}^{(e)}|} \sum_{\mathbf{x} \in \mathcal{O}_{\text{state}}^{(e)}} \min_{\mathbf{y} \in \mathcal{O}_{\text{goal}}^{(e)}} \|\mathbf{x} - \mathbf{y}\|_2 + \frac{1}{|\mathcal{O}_{\text{goal}}^{(e)}|} \sum_{\mathbf{y} \in \mathcal{O}_{\text{goal}}^{(e)}} \min_{\mathbf{x} \in \mathcal{O}_{\text{state}}^{(e)}} \|\mathbf{y} - \mathbf{x}\|_2. \tag{33}$$

[5] https://github.com/meta-pytorch/LeanRL

The average Chamfer distance over $N$ episodes is

$$\overline{\mathrm{CD}}(\pi;\mathcal{E}) = \frac{1}{N}\sum_{e=1}^{N} \mathrm{CD}^{(e)}(\pi). \tag{34}$$

**Task Success.** With respect to each environment, fix a threshold $\tau \in (0, 1]$. Define the episode-wise success indicator as

$$\mathrm{Success}^{(e)}(\pi) = \mathbb{I}\Big[\mathrm{Cov}^{(e)}(\pi) \geq \tau\Big]. \tag{35}$$

The success rate of policy $\pi$ on environment $\mathcal{E}$ is

$$\mathrm{SR}(\pi;\mathcal{E}) = \frac{1}{N}\sum_{e=1}^{N} \mathrm{Success}^{(e)}(\pi), \tag{36}$$

where $N$ is the number of evaluation episodes.

# F Expanded Related Work

Learning robust, generalizable policies for multi-object manipulation has progressed from leveraging hindsight to relabel goals in sparse-reward settings, to imposing structured, object-centric inductive biases on both perception and control. We organize the literature into two strands: (i) unsupervised object-centric representation learning, and (ii) object-centric reinforcement learning for goal-conditioned control.

## F.1 Object-Centric Representation Learning

Unsupervised object-centric learning seeks to decompose high-dimensional images into a set of latent entities that correspond to objects or parts, without pixel-level supervision. Early segmentation methods relied on hand-crafted features and graph-based clustering (e.g., Ncut (Shi & Malik, 2000), Felzenszwalb (Felzenszwalb & Huttenlocher, 2004)) or co-segmentation assumptions that require shared objects across images (Li et al., 2019; Choudhury et al., 2021), but did not aim to produce representations useful for downstream reasoning or control.

Modern approaches instead impose architectural biases that encourage the model to discover and bind distinct objects into separate latent codes. A dominant family uses slot-based attention. Slot Attention introduced a competitive attention mechanism that aggregates image features into a fixed set of exchangeable latent slots, each intended to capture one object's appearance and pose (Locatello et al., 2020). This framework is efficient and extends to video (Kipf et al., 2022), but suffers from several limitations: the number of slots must be fixed a priori, slots can entangle or split single objects, and performance degrades under diverse training distributions or varying object sizes (Jia et al., 2023). Subsequent work has proposed learnable slot initializations, bi-level optimization, and adaptive slot selection to improve stability and capacity (Jia et al., 2023), yet these methods still assume compact, slot-aligned representations that are ill-suited for modeling complex part–whole hierarchies.

An alternative line clusters learned visual features rather than optimizing slots end-to-end. Methods such as DINOSAUR and MaskCut first extract self-supervised features (e.g., from DINO) and then apply iterative attention or graph-based cuts to group similar regions into object masks (Seitzer et al., 2023; Wang et al., 2023a). OC-Net leverages feature connectivity and object-centric regularization to discover objects in complex scenes, achieving strong segmentation and downstream performance (Foo et al., 2023). However, these approaches typically produce flat, contrast-based segmentations that do not explicitly model how parts compose into multi-part objects, limiting their ability to support hierarchical reasoning or manipulation of structured entities.

More recent work explicitly targets part–whole compositionality. Approaches such as ECO-Net introduce graph-structured latent spaces and co-part discovery mechanisms to learn representations that factor objects into reusable parts and capture their relational structure (Foo et al., 2025). Other methods use cyclic walks between perceptual features and object entities to enforce part–whole consistency as a self-supervised signal (Wang et al., 2023b), improving disentanglement and semantic

object discovery without decoders. Despite these advances, most object-centric representations remain pre-trained and fixed when used in downstream RL, rather than being tuned end-to-end with the policy, which can hinder adaptation to task-specific dynamics and distribution shifts.

### F.2 Object-Centric Reinforcement Learning

Goal-conditioned reinforcement learning (GCRL) provides a framework for learning policies that can reach many possible goal states from a single policy (Kaelbling, 1993). Hindsight Experience Replay (HER) made GCRL practical in sparse-reward settings by relabeling trajectories so that achieved states become pseudo-goals, enabling learning even when the original goal is not reached (Andrychowicz et al., 2017). As tasks grow more complex and horizons lengthen, however, direct goal conditioning becomes inefficient, motivating hierarchical formulations in which high-level policies propose intermediate subgoals and low-level policies learn to achieve them (Levy et al., 2019). High-level policies propose latent or image subgoals, while low-level policies learn to reach them; recent offline methods such as HIQL employ action-free value functions and learn a high-level policy to propose latent subgoals from offline trajectories (Kostrikov et al., 2022; Park et al., 2023).

When observations are high-dimensional images, the choice of state representation critically affects sample efficiency and generalization. Holistic image-based GCRL methods, even with hierarchical subgoals, often struggle in multi-object settings and under distribution shift, as they must learn both perception and control jointly from pixels (Kostrikov et al., 2022; Park et al., 2023; Haramati et al., 2026). In contrast, object-centric RL methods first compress state and goal images into structured, object-level representations and then learn policies over these latent states. This separation enables more robust reasoning over multiple objects, better compositional generalization, and improved sample efficiency (Lin et al., 2023; Ferraro et al., 2025; Zhang et al., 2026).

ECRL demonstrated that compressing state–goal images into object-centric representations enables pixel-based multi-object manipulation under distribution shift (Haramati et al., 2024). EC-Diffuser extended this to the offline setting using a diffusion-based generative model to produce entity-centric behaviors from static datasets (Qi et al., 2025). HECRL further incorporated diffusion-based subgoal generation and offline IQL-style training to tackle long-horizon tasks (Haramati et al., 2026). Parallel work on object-centric world models—such as FOCUS, Dyn-O, and Latent Particle World Models—learns structured dynamics over object slots or particles, supporting planning and model-based control in manipulation domains (Ferraro et al., 2025; Wang et al., 2026; Daniel et al., 2026). Recent surveys highlight that object-centric RL can be broadly categorized into (i) learning better object-centric representations and (ii) designing policy and world-model architectures that exploit relational structure for compositional generalization.

Despite these advances, two key limitations persist. First, most pipelines use pre-trained, frozen object-centric representations that are not adapted end-to-end with the policy or world model, potentially misaligning perceptual inductive biases with task demands (Haramati et al., 2024; Qi et al., 2025). Second, existing subgoal-generation mechanisms, even when diffusion-based, often fail to generalize learned manipulation skills under significant distribution shift, especially when objects vary in number, appearance, or physical properties (Haramati et al., 2026). Addressing these gaps—by jointly learning adaptive object representations and robust policies—remains a central challenge for scalable, generalizable multi-object reinforcement learning.

## G AI USE STATEMENT

In this work, we used generative AI tools to polish portions of the manuscript text and to improve readability. We have not used generative AI tools for any tasks requiring mandatory disclosure, including generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating or proving mathematical claims, designing experiments or methodology, implementing methods, interpreting results, or performing data analysis. We have carefully reviewed and edited all AI-assisted text to ensure accuracy, clarity, and consistency with our intended meaning. We take full responsibility for the final content of this work, including all text, claims, and artifacts.

## H LIMITATIONS

There are still obstacles that must be overcome for successful application of our framework to the visual dynamic complexity of real world robotic manipulation. A natural next step would be to extend our method to even more efficient implementations of object discovery and representation, especially for applications that require real-time planning and refinement. End-to-end training of our framework could also be further optimized for large-scale offline training and possibly real-time adaptation without catastrophic forgetting. Lastly, real-world manipulation scenes involving multi-object manipulation under varying illumination, viewpoints, and occlusions present higher visual complexity than the simulated environments considered here, and reliably bridging this simulation-to-reality gap remains an open problem.

## I REPRODUCIBILITY STATEMENT

We are committed to enabling the reproducibility of our work. The paper and appendix provide sufficient implementation and evaluation details, and a standalone implementation of COMPACT is included in the supplementary material. We will release the full code upon acceptance of the paper.

## J ETHICS STATEMENT

Our analysis has no immediate negative impact on general society. To the best of our knowledge, none of the environments and datasets used in this study contain personally identifiable information or offensive content. Our work advances methods for offline Goal-Conditioned Reinforcement Learning (GCRL) and is evaluated entirely in simulated environments, with no involvement of human subjects or privacy concerns. As with any model that performs visual reasoning and manipulation planning, potential negative societal impacts, such as in autonomous systems or surveillance-related applications, cannot be fully excluded in future research. However, significant open challenges remain before reliable deployment in real-world settings, making direct malicious application currently unlikely.