TarGF: Learning Target Gradient Field for Object Rearrangement

Mingdong Wu[*1,3], **Fangwei Zhong**[*2,3], **Yulong Xia**[1], **Hao Dong**[1,4]
[1] Center on Frontiers of Computing Studies, School of Computer Science, Peking University
[2] School of Intelligence Science and Technology, Peking University
[3] Beijing Institute for General Artificial Intelligence (BIGAI)
[4] Peng Cheng Laboratory
{wmingd, zfw, hao.dong}@pku.edu.cn

Abstract

Object Rearrangement is to move objects from an initial state to a goal state. Here, we focus on a more practical setting in object rearrangement, *i.e.*, rearranging objects from shuffled layouts to a normative target distribution without explicit goal specification. However, it remains challenging for AI agents, as it is hard to describe the target distribution (goal specification) for reward engineering or collect expert trajectories as demonstrations. Hence, it is infeasible to directly employ reinforcement learning or imitation learning algorithms to address the task. This paper aims to search for a policy only with a set of examples from a target distribution instead of a handcrafted reward function. We employ the score-matching objective to train a ***Tar**get **G**radient **F**ield (TarGF)*, indicating a direction on each object to increase the likelihood of the target distribution. For object rearrangement, the TarGF can be used in two ways: 1) For model-based planning, we can cast the target gradient into a reference control and output actions with a distributed path planner; 2) For model-free reinforcement learning, the TarGF is not only used for estimating the likelihood-change as a reward but also provides suggested actions in residual policy learning. Experimental results in ball rearrangement and room rearrangement demonstrate that our method significantly outperforms the state-of-the-art methods in the quality of the terminal state, the efficiency of the control process, and scalability. The code and demo videos are on https://sites.google.com/view/targf.

1 Introduction

As shown in Fig. 1, we consider object rearrangement *without explicit goal specification* [1, 2], where the agent is required to manipulate a set of objects from an initial layout to a normative distribution. This task is taken for granted by humans [3] and widely exists in our daily life, such as tidying up a table [4], placing furniture [5] , and sorting parcels [6].

However, compared with conventional robot tasks [7, 8, 9, 10, 11], there are two critical challenges in our setting: 1) Hard to clearly define the target state (goal), as the target distribution is diverse and the patterns are difficult to describe in the program, *e.g.*, how to evaluate the *tidiness*

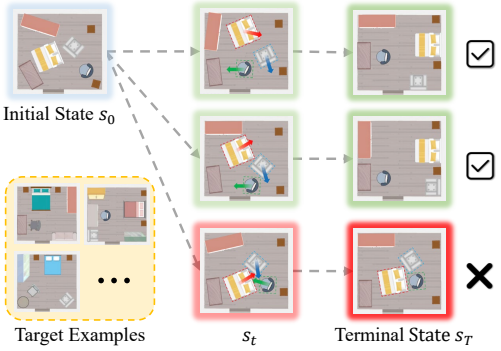


Figure 1: Our goal: Learning to rearrange objects **without explicit goal specification**.

* indicates equal contribution

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

quantitatively? As a result, it is difficult to manually design a reward/objective function, which is essential to modern control methods, *e.g.*, deep reinforcement learning (DRL) [12]. 2) In the physical world, the agents also need to find an executable path to reach the target state efficiently and safely, *i.e.*, finding a short yet collision-free path to reach the target distribution. To this end, the agents are required to jointly learn to evaluate the quality of the state (reward) and move objects to increase the quality efficiently. In other words, the agents should answer the two questions: *What kind of layouts does the user prefer*, and *How to manipulate objects to meet the target distribution*?

Thus, it is necessary to search for a policy without dependence on the explicit goal state and shaped reward. Imitation learning (IL) can learn a reward function [13, 14, 15] or learn a policy [16, 17, 18, 19] from the expert's supervision, *e.g.*, demonstrating expected trajectories. However, collecting a large number of expert trajectories is expensive. Recently, example-based reinforcement learning (RL) [20, 21] has tried to reduce the reliance on expert trajectories, *i.e.*, learning a policy only with examples of successful outcomes. However, it is still difficult to learn a classifier to provide accurate reward signals for learning in the case of high-dimensional state and action space, as there are many objects to control in object rearrangements.

In this work, we separate the task into two stages: 1) learning a score function to estimate the target-likelihood-change between adjacent states, and 2) learning/building a policy to rearrange objects with the score function. Inspired by the recent advances in score-based generative models [22, 23, 24, 25, 26], we employ the denoising score-matching objective [27] to train a *target score network* for estimating the *target gradient field*, *i.e.*, the gradient of the log density of the target distribution. The target gradient field can be viewed as object-wise guidance that provides a pseudo velocity for each object, pointing to regions with a higher density of the target distribution, *e.g.*, tidier configurations. To rearrange objects in the physical world, we further demonstrate two ways to leverage the target gradient field in control. 1) We integrate the gradient-based guidance with a model-based path planner to rearrange objects without collision. 2) We further derive a reward function for reinforcement learning by estimating the likelihood change via the target gradient. With the TarGF-based reward and a TarGF-guided residual policy network [28], we can train an effective RL-based policy.

In experiments, we evaluate the effectiveness of our methods on two typical rearrangement tasks: *Ball Rearrangement* and *Room Rearrangement*. First, we introduce three metrics to analyse the quality of the terminal state and the efficiency of the control process. In *Ball Rearrangement*, we demonstrate that both of our control methods can efficiently reach a terminal configuration of high quality and diversity. We also observe that our methods can find a path with fewer collisions and shorter lengths to converge compared with learning-based baselines [14, 21] and planning-based baselines [29]. In *Room Rearrangement*, we further evaluate the feasibility of our methods in complex scenes where there are multiple heterogeneous objects.

Our contributions are summarised as follows:

- We introduce a novel score-based framework that estimates a *target gradient field (TarGF)* via denoising score matching for object rearrangement.
- We point out two usages of the target gradient field in object rearrangement: 1) providing guidance in moving direction; 2) deriving a reward for learning.
- We conduct experiments to demonstrate that both traditional planning and RL methods can benefit from the target gradient field and significantly outperform the state-of-the-art methods in sample diversity, realism and safety.

2 Related Works

2.1 Object and Scene (Re-)Arrangement

The object (re-)arrangement is a long-studied problem in robotics community [2, 30, 31] and the graphics community [32, 33, 34, 35, 36]. However, most of them focus on manually designing rules/energy functions to find a goal [2, 30, 31] or synthesising a scene configuration [33, 34, 35, 36] that satisfies the preference of the user. The most recent work [4] tries to learn a GNN to output an arrangement tailored to human preferences. However, they neglect the physical process of rearrangement. Hence, the accessibility and transition cost from the initial and target states are not guaranteed. Considering the physical interaction, the recent works on scene rearrangement [37, 38] mitigate the goal specification problem by providing *a goal scene* to the agent, making the reward shaping feasible. Concurrent works [39, 40] also notice the necessity of automatic goal inference for

tying rooms and exploit the commonsense knowledge from Large Language Model (LLM) or memex graph to infer rearrangements goals when the goal is unspecified. Another concurrent work [41] aims at discovering generalisable spatial goal representations via graph-based active reward learning for 2D object rearrangement. In this paper, we focus on a more practical setting of rearrangement: *how to estimate the similarity (i.e., the target likelihood) between the current state and the example sets and manipulate objects to maximise it*. Here, only a set of positive target examples are required to learn the gradient field, rather than the prior knowledge about specific scenarios.

2.2 Learning without Reward Engineering

Without hand-engineered reward, previous works explore algorithms to learn a control policy from expert demonstrations via behavioural cloning [16, 17, 18, 13] or inverse reinforcement learning [42, 15, 43, 44]. Imitation learning (IL) [16, 17, 18] aims to directly learn a policy by cloning the behaviour from the expert trajectories. The inverse reinforcement learning (IRL) [42, 15, 43, 44] tries to learn a reward function from data for subsequent reinforcement learning. Considering the difficulty in collecting expert trajectories, there are some example-based methods proposed to learn a policy only with a set of successful examples [20, 14, 21]. The most recent work is RCE [21] which directly learns a recursive classifier from successful outcomes and interactions for policy optimisation. We also try to develop an example-based control method for rearrangement with a set of examples. Our method can learn the target gradient field from the examples, without any additional efforts in reward engineering. The target gradient field can provide meaningful reward signals and action guidance for reinforcement learning and traditional path planners. To the best of our knowledge, such usage of gradient field is not explored in previous works.

3 Problem Statement

We aim to learn a policy that moves objects to reach states close to a target distribution without relying on reward engineering. Similar to the *example-based control* [21], the grounded *target distribution* $p_{tar}(\mathbf{s})$ is unknown to the agent, and the agent is only given a set of *target examples* $S^* = \{\mathbf{s}^*\}$ where $\mathbf{s}^* \sim p_{tar}(\mathbf{s})$. In practice, these examples can be provided by the user without accessing the dynamics of the robot. The agent starts from an initial state $\mathbf{s}_0 \sim p_0(\mathbf{s}_0)$, where N objects are randomly placed in the environment. At time step t, the agent takes action $\pi(\mathbf{a}_t|\mathbf{s}_t)$ imposed on the objects to reach a next state with dynamics $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. Here, the goal of the agent is to search for a policy $\pi^*(\mathbf{a}_t|\mathbf{s}_t)$ that maximises the *discounted log-target-likelihood* of the future states:

$$\pi^* = \arg\max_{\pi} E_{\rho(\mathbf{s}_0),\tau\sim\pi}\left[\sum_{\mathbf{s}_t\in\tau} \gamma^t \log p_{tar}(\mathbf{s}_t)\right] \tag{1}$$

where $\gamma \in (0, 1]$ denotes the discount factor. Notably, we assume that $p_{tar}(\mathbf{s}) > 0$ everywhere since we can perturb the original data with a tiny noise (*e.g.*, $\mathcal{N}(0, 0.00001)$) to ensure the perturbed density is always positive. [22] also used this trick to tackle the manifold hypothesis issue. This objective reveals three challenges: **1) Inaccessibility problem:** The grounded target distribution p_{tar} in Eq. 1 is inaccessible. Thus, we need to learn a function to approximate the *log-target-likelihood* for policy search. **2) Sparsity problem:** The log-target-likelihood is sparse in the state space due to the *Manifold Hypothesis* mentioned in [22]. Hence, even if we have access to the target distribution p_{tar}, it is still difficult to explore the high-density region. **3) Adaptation problem:** The dynamics $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ is also inaccessible. To maximise the cumulative sum in Eq. 1, the agent is required to adapt to the dynamics to efficiently increase the target likelihood.

To address these problems, we partition the task into 1) learning to estimate the log-target-likelihood (reward) of the state and 2) learning/building a policy to rearrange objects to adapt to the dynamics.

4 Method

4.1 Motivation

Estimating the *gradient* of the log-target-likelihood $\nabla_{\mathbf{s}} \log p_{tar}(\mathbf{s})$ can tackle the first two problems mentioned in Sec. 3: For the *inaccessibility* problem, we can approximate the likelihood increment

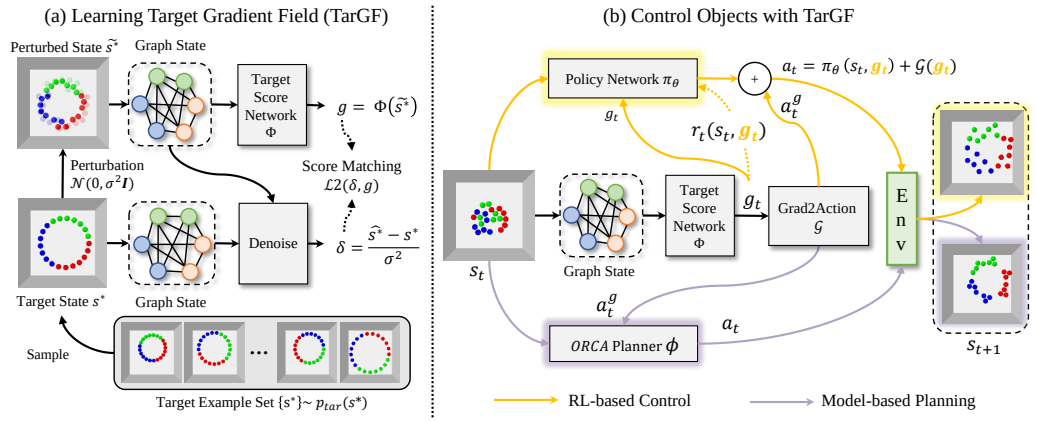


Figure 2: An overview of our method. (a) We train the target score network via score matching. The target examples are first perturbed by Gaussian noise. The target score network is forced to match the denoising direction of the perturbation. (b) Our framework is based on the trained target score network. In the model-free setting, the target score network provides exploration guidance and reward estimation for RL. In the model-based setting, the TarGF provides reference velocities for a model-based planner based on ORCA. The planner then outputs collision-free velocities for objects.

between two adjacent states by the first-order Taylor expansion $\log p_{tar}(\mathbf{s}_{t+1}) - \log p_{tar}(\mathbf{s}_t) \approx \langle \nabla_{\mathbf{s}} \log p_{tar}(\mathbf{s}_t), \mathbf{s}_{t+1} - \mathbf{s}_t \rangle$. This helps to derive a surrogate objective (*i.e.*, Eq. 4) of Eq. 1.

For the *sparsity* problem, we can leverage the gradient field to help with exploration since the gradient $\nabla_{\mathbf{s}} \log p_{tar}(\mathbf{s})$ indicates the fastest direction to increase the target likelihood. To estimate $\nabla_{\mathbf{s}} \log p_{tar}(\mathbf{s})$, we employ the score-matching [23], a novel generative model that achieved impressive results in many research areas recently[24, 26, 25], to train a *target score network* $\mathbf{\Phi}_{tar} : \mathcal{S} \to \mathcal{S}$.

To address the *adaptation* problem, we further demonstrate two approaches to incorporate the trained target gradient field $\mathbf{\Phi}_{tar}$ with control algorithms for object rearrangement: 1) For the model-based setting, our framework casts the target gradient into a reference control and outputs an action with a distributed path planner. 2) For the model-free setting (reinforcement learning), we leverage the target gradient to estimate reward and provide suggested action in residual policy learning [28].

In the followings, we will first introduce how to train the target gradient field $\mathbf{\Phi}_{tar}$ via score-matching in Sec. 4.2 and then introduce our rearrangement framework under the model-based and model-free settings in Sec. 4.3 and Sec. 4.4 respectively.

4.2 Learning the Target Gradient Field from Examples

The target score network $\mathbf{\Phi}_{tar}$ is the **core module** of our framework, which aims at estimating the *score function* (*i.e.*the gradient of log-density) of the target distribution $\nabla_{\mathbf{s}} \log p_{tar}(\mathbf{s})$.

To train the target score network, we adopt the Denoising Score-Matching (DSM) objective proposed by [27], which can guarantee a reasonable estimation of the $\nabla_{\mathbf{s}} \log p_{tar}(\mathbf{s})$. In training, we pre-specify a noise distribution $q_\sigma(\tilde{\mathbf{s}}|\mathbf{s}) = \mathcal{N}(\tilde{\mathbf{s}}; \mathbf{s}, \sigma^2 I)$. Then, DSM matches the output of the target score network with the score of the perturbed target distribution $q_\sigma(\tilde{\mathbf{s}}) = \int q_\sigma(\tilde{\mathbf{s}}|\mathbf{s}) p_{tar}(\mathbf{s}) d\mathbf{s}$:

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathbf{s}}|\mathbf{s}), p_{tar}(\mathbf{s})}\left[||\mathbf{\Phi}_{tar}(\tilde{\mathbf{s}}) - \nabla_{\tilde{\mathbf{s}}} \log q_\sigma(\tilde{\mathbf{s}}|\mathbf{s})||_2^2\right] = \frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathbf{s}}|\mathbf{s}), p_{tar}(\mathbf{s})}\left[||\mathbf{\Phi}_{tar}(\tilde{\mathbf{s}}) - \frac{\mathbf{s} - \tilde{\mathbf{s}}}{\sigma^2}||_2^2\right] \quad (2)$$

The DSM objective guarantees the optimal score network satisfies $\mathbf{\Phi}^*_{tar}(\mathbf{s}) = \nabla_{\mathbf{s}} q_\sigma(\mathbf{s})$ almost surely. When σ is small enough, we have $\nabla_{\mathbf{s}} q_\sigma(\mathbf{s}) \approx \nabla_{\mathbf{s}} \log p_{tar}(\mathbf{s})$, so that $\mathbf{\Phi}^*_{tar}(\mathbf{s}) \approx \nabla_{\mathbf{s}} \log p_{tar}(\mathbf{s})$.

In practice, we adopt a variant of DSM proposed by [23], which conducts DSM under different noise scales simultaneously. In this way, we can efficiently try different noise scales after only one training. The target score network is implemented as a graph-based network $\mathbf{\Phi}_{tar}(\mathbf{s}, t)$ which is conditioned on a noise level $t \in (0, 1]$. The input states $= [\mathbf{s}^1, \mathbf{s}^2, ..., \mathbf{s}^N]$ is constructed as a fully-connected graph that takes each object's state (*e.g.*position, orientation) $\mathbf{s}^i \in \mathbb{R}^{f_s}$ and static properties (*e.g.*category, bounding-box) $\mathbf{p}^i \in \mathbb{R}^{f_p}$ as the node feature $[\mathbf{s}^i, \mathbf{p}^i] \in \mathbb{R}^{f_s + f_p}$. The input graph is pre-processed

by linear feature extraction layers and then passed through several Edge-Convolution layers. After message-passing, the output on the i-th node serves as the component of the target gradient on the i-th object's state $\mathbf{\Phi}^i_{tar}(\mathbf{s}) \in \mathbb{R}^{f_s}$. We defer structure details to Appendix. B.

4.3 Model-based Planning with the Target Gradient Field

The target gradient $\mathbf{g} = \mathbf{\Phi}_{tar}(\mathbf{s})$ can be translated into a *gradient-based action* $\mathbf{a}^g = \mathcal{G}(\mathbf{g})$, where $\mathcal{G}$ denotes the gradient-to-action translation. A gradient over the state space can be viewed as velocities imposed on each object. For instance, if the state of each object is a 2-dimensional position $\mathbf{s}_i = [x, y]$, then the target gradient component on the i-th object can be viewed as linear velocities on two axes $\mathbf{\Phi}_{tar}(\mathbf{s})^i = [v_x, v_y]$. When the action space is also velocities, we can construct $\mathbf{a}^g$ by simply projecting the target gradient into the action space $\mathbf{a}^g = \mathcal{G}(\mathbf{g}) = \frac{v_{max}}{||\mathbf{g}||_2} \cdot \mathbf{g}$, where the v_{max} denotes the speed limit. Following the gradient-based action, objects will move towards directions that may increase the likelihood of the next state, which meets the need for the rearrangement task.

However, the gradient-based action will potentially lead objects to collide with each other since the target gradient cannot infer the environment constraints from only target examples.

To adapt the gradient-based action to the environment dynamics, we incorporate the target gradient field with a model-based off-the-shelf collision-avoidance algorithm *ORCA*. The ORCA planner ϕ takes the gradient-based action $\mathbf{a}^g$ as reference velocities and then outputs the collision-free velocities $\mathbf{a} = \phi(\mathbf{a}^g)$ with the least modification to the reference velocities. In this way, the adapted action can lead objects to move towards a higher likelihood region efficiently and safely. For more details of the ORCA planner, we defer to Appendix B.2.

4.4 Learning Policy with the Target Gradient Field

The model-based approach to 'correct' the target gradient assumes the objects are of circular shape. This limits the scope of this method when objects are of non-circular shape (*e.g.*, furniture). To this end, we further propose a *model-free approach* based on reinforcement learning (RL) for object rearrangement, where the agent needs to adapt the dynamics via online interactions.

To tackle the inaccessibility problem mentioned in Sec. 3, we first derive an equivalent $J^*(\pi)$ of the original objective in Eq. 1 by subtracting a constant from the original objective:

$$\begin{aligned} J(\pi) \iff & \mathbb{E}_{\rho(\mathbf{s}_0),\tau\sim\pi}[\sum_{\mathbf{s}_t\in\tau} \gamma^t \log p_{tar}(\mathbf{s}_t)] - \underbrace{\frac{\mathbb{E}_{\rho(\mathbf{s}_0)}[\log p_{tar}(\mathbf{s}_0)]}{1-\gamma}}_{constant \quad C} \\ = & \mathbb{E}_{\rho(\mathbf{s}_0),\tau\sim\pi}[\sum_{1\le t\le T} \gamma^t \sum_{1\le k\le t} [\log p_{tar}(\mathbf{s}_k) - \log p_{tar}(\mathbf{s}_{k-1})]] \overset{def}{=} J^*(\pi) \end{aligned} \tag{3}$$

Further, we derive a surrogate objective $\hat{J}(\pi)$ to approximate the $J^*(\pi)$. We notice that in most physical simulated tasks, the distance between two adjacent states $||\mathbf{s}_t - \mathbf{s}_{t-1}||_2$ is quite small, which inspires us to approximate the log-likelihood-change of the adjacent states using the Taylor expansion:

$$J^*(\pi) \approx \mathbb{E}_{\rho(\mathbf{s}_0),\tau\sim\pi}[\sum_{1\le t\le T} \gamma^t \underbrace{\sum_{1\le k\le t} \langle \mathbf{\Phi}_{tar}(\mathbf{s}_{k-1}), \mathbf{s}_k - \mathbf{s}_{k-1}\rangle}_{r_t}] \overset{def}{=} \hat{J}(\pi) \tag{4}$$

In this way, we can optimise the surrogate objective $\hat{J}(\pi)$ by assigning $\sum_{1\le k\le t} \langle \mathbf{\Phi}_{tar}(\mathbf{s}_t), \mathbf{s}_k - \mathbf{s}_{k-1}\rangle$ as the immediate reward. In practice, we simply assign the last term of the cumulative summation as the immediate reward $r_t = \langle \mathbf{\Phi}_{tar}(\mathbf{s}_t), \mathbf{s}_t - \mathbf{s}_{t-1}\rangle$, and this version is shown to be the most effective.

To further tackle the sparsity problem, our idea is to build a residual policy π_θ upon the gradient-based action $\mathbf{a}^g_t$ mentioned in Sec. 4.3, as $\mathbf{a}^g_t$ helps to explore the regions with a high target likelihood:

$$\underbrace{\mathbf{a}_t = \pi_\theta(\mathbf{s}_t, \mathbf{g}_t) + \mathbf{a}^g_t}_{output\,action}, \quad \underbrace{\mathbf{a}^g_t = \mathcal{G}(\mathbf{g}) = \frac{v_{max}}{||\mathbf{g}_t||_2} \cdot \mathbf{g}_t}_{gradient-based\,action}, \quad \underbrace{\mathbf{g}_t = \mathbf{\Phi}_{tar}(\mathbf{s}_t)}_{target\,gradient} \tag{5}$$

This approach is similar to the residual policy learning proposed by [28]. The $\mathbf{a}_t^g$ serves as an initial policy, and the residual policy network π_θ takes the target gradient $\mathbf{g}_t$ and the current state $\mathbf{s}_t$ as input and outputs $\pi(\mathbf{s}_t, \mathbf{g}_t)$ to 'correct' the $\mathbf{a}_t^g$. The agent finally outputs a 'corrected' action $\mathbf{a}_t = \pi(\mathbf{s}_t, \mathbf{g}_t) + \mathbf{a}_t^g$. In this way, the agent can not only benefit from the efficient exploration but also have less training burden as it 'stands on the giant's shoulder' (*i.e.*the gradient-based action $\mathbf{a}_t^g$).

We employ the multi-agent Soft-Actor-Critic(SAC) [45] as the reinforcement learning algorithm backbone. Each object is regarded as an agent, and each agent can communicate with all the other agents. At each time step t, the reward for the i-th agent is $r_t = r_t^{tar} + \lambda * r_t^i$, where $r_t^{tar} = \sum_{1 \le k \le t} \langle \mathbf{\Phi}_{tar}(\mathbf{s}_t), \mathbf{s}_k - \mathbf{s}_{k-1} \rangle$ is a centralised target likelihood reward, $\lambda > 0$ is a hyperparameter and $r_t^i = \sum_{j \neq i} c_{i,j}$ is a decentralised collision penalty where $c_{i,j}$ is a collision penalty. Specifically, when a collision is detected between object i and j, $c_{i,j} = -1$, otherwise $c_{i,j} = 0$.

5 Experiment Setups

5.1 Environment

We design two object rearrangement tasks without explicit goal specification for evaluation: *Ball Rearrangement* and *Room Rearrangement*, where the former uses controlled environments for better numerical analysis and the latter is built on a real dataset with implicit target distribution.

Ball Rearrangement includes three environments with increasing difficulty: *Circling*, *Clustering*, and the hybrid of the first two, *Circling + Clustering*, as shown in Fig. 3. There are 21 balls of the same geometry in each task. In *Clustering* and *Circling+Clustering*, the balls are divided equally into three colours. The goals of the three tasks are as follows: *Circling* requires all balls to form a circle. The centre of the circle can be anywhere in the environment; *Clustering* requires all balls to form into three clusters by colour. The joint centre of each cluster has two choices (*i.e.*, red-green-blue or red-blue-green clockwise); *Circling + Clustering* requires all balls to form into a circle where balls of the same type are adjacent. To slightly increase the complexity of the physical dynamics, we enlarge the lateral friction coefficient of each ball and make each ball with different physical properties, *e.g.*, different friction coefficients and masses. The target examples of each task are sampled from an explicit process. Thus, we can define a 'pseudo likelihood' on a given state according to the sampling process to measure the similarity between the state and the target distribution.

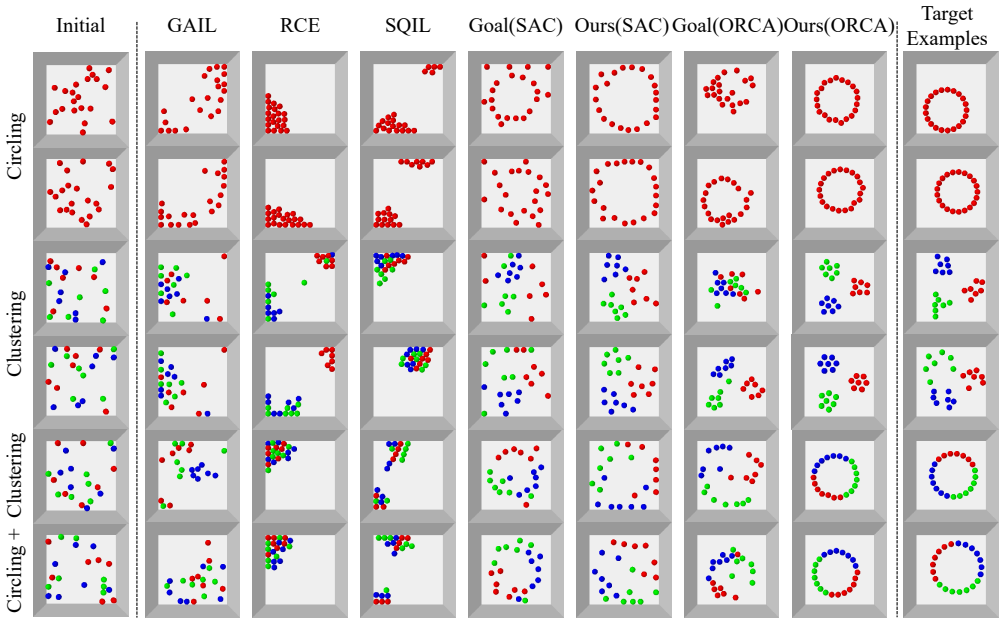


Figure 3: Starting from the same initial states on the far left, we demonstrate the rearrangement results of different methods. On the far right, we demonstrate some target examples of each task.

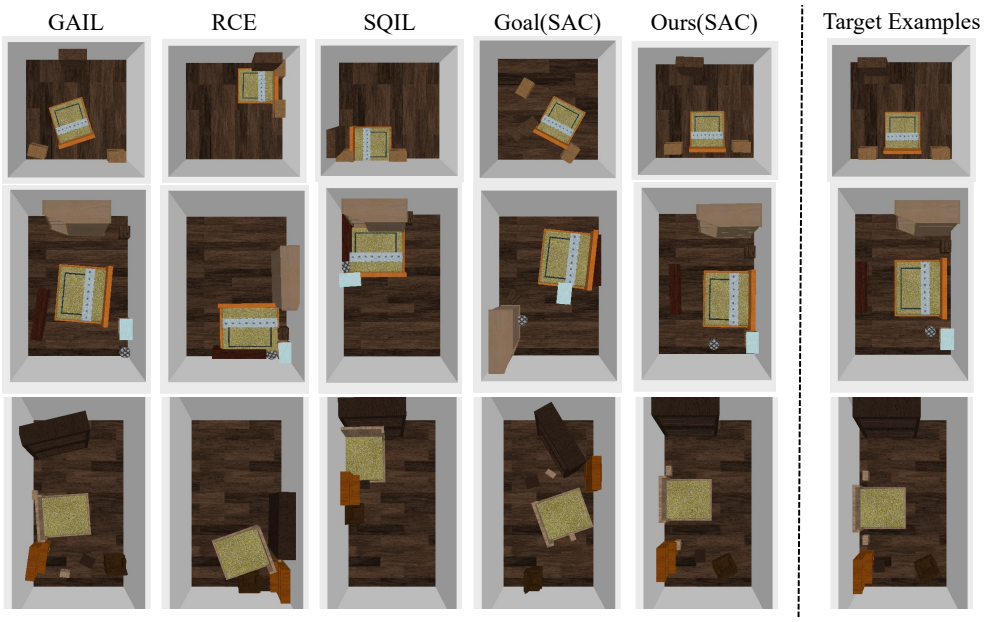


Figure 4: Qualitative results on room rearrangement. We obtain eight rearrangement results of each method starting from the same set of eight initial states. Then we demonstrate the rearrangement result that is closest to the target example for each method.

Room Rearrangement is built on a more realistic dataset, 3D-FRONT [46]. After cleaning the data, we get a dataset with 839 rooms, 756 for training, *e.g.*, target examples, and 83 for testing. In each episode, we sample a room from the test split and shuffle the room via a Brownian Process at the beginning to get an initial state. The agent can assign angular and linear velocities for each object in the room at each time step. To tidy up the room, the agent has to learn implicit knowledge learned from the target examples, as shown in Fig. 4. More details are in the Appendix A.

5.2 Evaluation

For both baseline comparison and ablation studies, we collect a fixed number of trajectories $T = \{\tau_i\}$ starting from the same initial states for each of 5 different random seeds. For each random seed, we collect 100 and 83×8 (8 initial states for each room) trajectories for ball and room rearrangement respectively. Then we calculate the following metrics on the trajectories, where the PL is only reported in ball rearrangement. More details of evaluation are in the Appendix D.

Pseudo-Likelihood (PL) measures the efficiency of the rearrangement process and the sample quality of the results. We specify a *pseudo-likelihood function* $\mathbf{F}_{proxy} : \mathcal{S} \rightarrow \mathbb{R}^+$ for each environment that measures the similarity between a given state and the target examples. For each time step t, we report the mean PL over trajectories $\mathbb{E}_{\tau \sim T}[\mathbf{F}_{proxy}(\mathbf{s}_t)]$ and the confidence interval over 5 random seeds. We do not report PL on room rearrangement as it is hard to program human preferences.

Coverage Score (CS) measures the diversity and fidelity of the rearrangement results (*i.e.*the terminal states $S_T = \{\mathbf{s}_T\}$ of the trajectories) by calculating the Minimal-Matching-Distance (MMD) [47] between S_T and a fixed set of examples S_{gt} from p_{tar}: $\sum_{\mathbf{s}_{gt} \in S_{gt}} \min_{\mathbf{s}_T \in S_T} ||\mathbf{s}_{gt} - \mathbf{s}_T||$. If the rearrangement results of a method miss some modes in the ground truth example set, the coverage score will increase so as to hurt the performance.

Averaged Collision Number (ACN) reflects the safety and efficiency of the rearrangement process since object collision will lead to object blocking and deceleration. ACN is calculated as $\sum_{\tau \in T} \sum_{\mathbf{s}_t \in \tau} \mathbf{c}_t$, where $\mathbf{c}_t$ denotes the total collision number at time step t.

5.3 Baselines

We compare our framework, *i.e.*, *Ours (ORCA) and Ours (SAC)*, with planning-based baselines and learning-based baselines. Besides, we design heuristic baselines serving as the upper bounds for ball

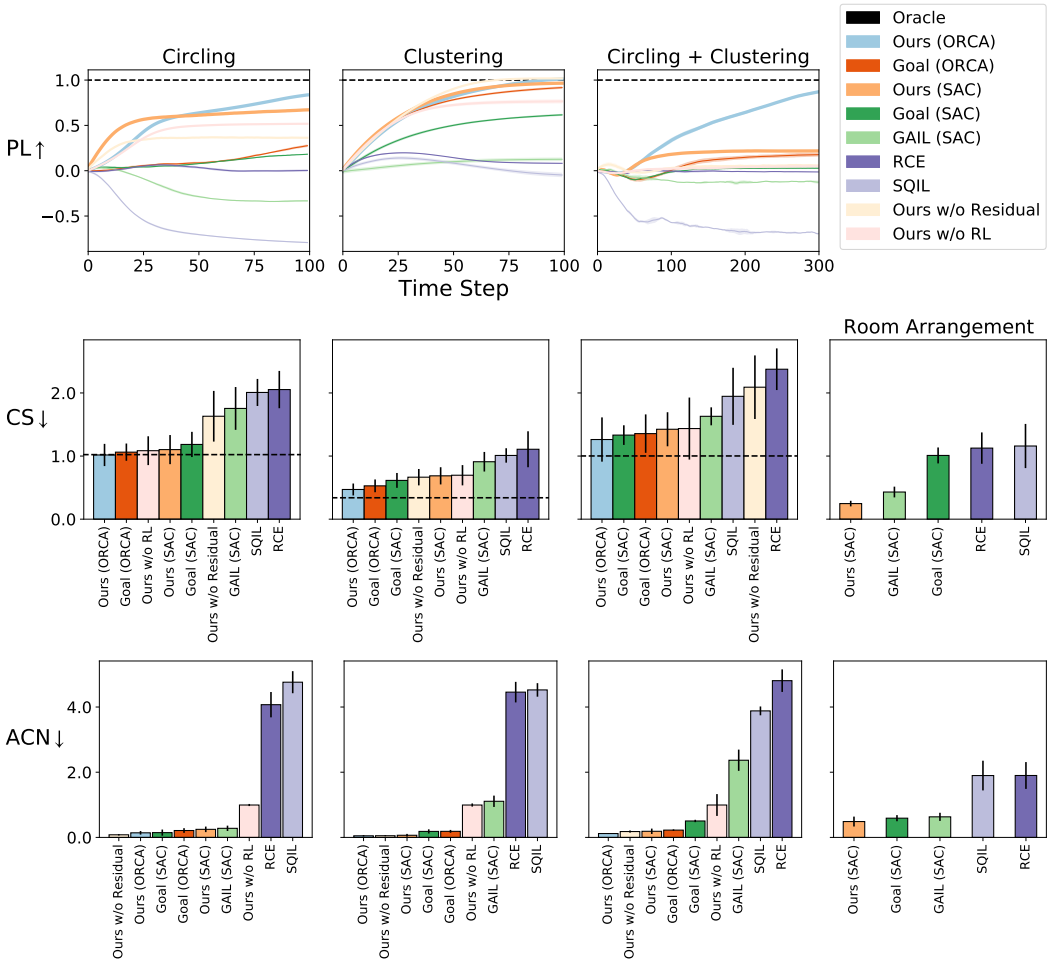


Figure 5: Quantitative results on ball rearrangement. From top to down, we report pseudo-likelihood (PL), coverage score (CS), and averaged collision number (ACN). The PL curves and CS bars of each task are normalised by the oracle's performance of this task. For CS and ACN, we report the mean and confidence interval over five random seeds.

rearrangement tasks. All the reinforcement learning-based methods share the same network design, capacity and hyperparameters. More details of baselines are in Appendix. C.

Leaning-based Baselines: *GAIL:* A classical inverse RL method that trains a discriminator as the reward function. *RCE:* A strong example-based reinforcement learning method proposed recently. *SQIL:* A competitive example-based imitation learning method. *Goal-RL:* An 'open-loop' method that first generates a goal at the beginning of the episode via a VAE trained from the target examples and then trained a goal-conditioned policy to reach the proposed goal via RL.

Planning-based Baselines: *Goal (ORCA):* This method first generates a goal via the same VAE as in *Goal (SAC)*. Then the agent assigns an action pointing to the goal as reference control and adjusts the preference velocity using ORCA, similar to Sec. 4.3.

Heuristic-based Baseline: *Oracle:* We slightly perturb the samples from the target distribution and take these samples as the rearrangement results. The rearrangement results of this method will be so close to the target distribution yet different from the target examples. Hence, we take this method as a reference for normalising the PL curve and CS bars.

6 Result Analysis

6.1 Baseline Comparison

Ball Rearrangement: We present the quantitative results on ball rearrangement tasks in the first three columns of Fig. 5. As shown on the top row of Fig. 5, the PL curves of *Ours (SAC)* and

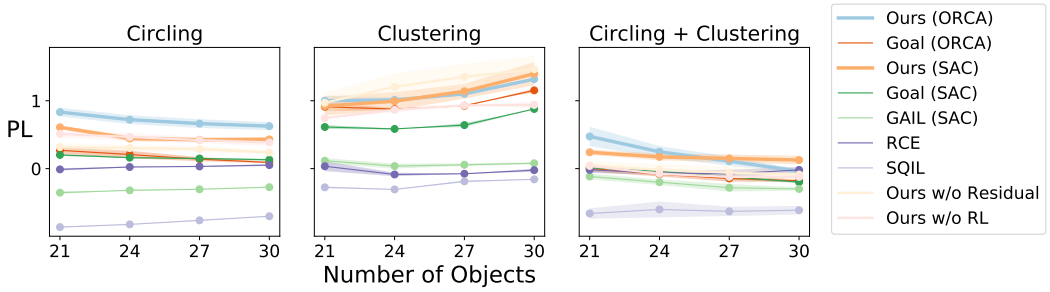


Figure 6: **Scalability Anaslysis**: Each policy is trained on environments with 3×7 balls and is zero-shot transferred to environments with different ball numbers.

Ours (ORCA) are significantly higher and steeper than all the baselines in all three tasks, which shows the effectiveness of our method in arranging objects towards the target distribution efficiently. In the middle row of Fig. 5, *Ours (ORCA)* achieves the best performance in CS across all three tasks. *Goal (SAC)* outperforms *Ours (SAC)* on *Clustering* and *Circling+Clustering* since the goal is proposed by a generative model that can naturally generate diverse and realistic goals. From the qualitative results shown in Fig. 3, our methods, especially *Ours (ORCA)*, also outperform the baselines in realism. Our methods also achieve the lowest ACN in most cases, except that *Goal (SAC)* is slightly better than *Ours (SAC)* in *Circling*.

The *Goal (SAC)* is the strongest baseline yet loses to our framework in efficiency since the generated goals are far away from the initial state. We validate this problem by comparing the averaged path length of the *Goal (SAC)* with ours in Appendix. E.3. Results show that our methods converge in a shorter path than goal-based baselines across all three tasks. Besides, the *Goal (SAC)* suffers from invalid goal proposals. As shown in Fig. 3, the *Goal (SAC)* and *Goal (ORCA)* can rearrange objects in the right trend, but the objects may block each other's way. This is because the goals generated by the VAE are not guaranteed to be valid (*e.g.*the balls may overlap with each other) so the agent cannot reach the invalid goal. We demonstrate this problem in Appendix. E.4.

The classifier-based reward learning method, *GAIL*, *RCE*, and *SQIL*, basically fails in most tasks, as shown in Fig. 3 and Fig. 5. This is due to the over-exploitation problem that commonly exists in classifier-based reward learning methods. We validate this in Appendix E.2.

Room Rearrangement: We only compare *Ours (SAC)* with learning-based baselines because it is infeasible to employ the ORCA planner in room rearrangement since all the furniture is non-circular. As shown in Fig. 4, our method can obtain more realistic rearrangement results compared with baselines. As shown in the last column of Fig. 5, the coverage score of *Ours (SAC)* is the lowest and almost half of that of *Goal (SAC)*. Meanwhile, *Ours (SAC)* achieves the lowest ACN compared with baselines. These results prove our method can also work on a more complex task and can generalise well to unseen conditions.

6.2 Ablation Studies and Analysis

We conduct ablation studies on ball rearrangement tasks to investigate: 1) The effectiveness of the *exploration guidance* provided from the target gradient field and 2) The necessity of combining the target gradient field with *control algorithms* to adapt to the environment dynamics. To this end, we construct an ablated version named *Ours w/o Residual* that drops the residual policy learning and another one named *Ours w/o RL* that only takes gradient-based action $\mathbf{a}^g$ as policy.

Ours w/o Residual: The CS bars in Fig. 5 shows that *Ours (SAC)* significantly outperforms the *Ours w/o Residual* in CS, which means the agent is easy to over-exploit few modes with high density. We also show qualitative results in Appendix E.1 that without the residual policy learning, the objects are usually arranged into a single pattern.

Ours w/o RL: The PL curves in Fig. 5 shows that without the control algorithm serving as the backbone, the rearrangement efficiency of *Ours w/o RL* is significantly lower than *Ours (SAC)* and *Ours (ORCA)*. Meanwhile, the ACN of *Ours w/o RL* is significantly larger than *Ours (SAC)* and *Ours (ORCA)*, which indicates the gradient-based action may cause severe collisions without adapting to the environment dynamics.

Scalability: We further evaluate the scalability of our methods by zero-shot transferring the learned model to rearrange different numbers of objects. To be specific, we train all the policies in the ball environment with $3 \times 7 = 21$ balls. During the testing phase, we directly transfer each policy to environments with increased ball numbers, i.e., 3×8, 3×9, and 3×10, without any fine-tuning. In Fig. 6.2, we report the averaged PL increment from the initial to target states of each method. Results show that our method still outperforms baselines under different numbers of balls.

7 Conclusion

In this study, we first analyse the challenges of object rearrangement without explicit goal specification and then incorporate a target gradient field with either model-based planning or model-free learning approaches to achieve this task. Experiments demonstrate our effectiveness by comparisons with learning-based and planning-based baselines.

Limitation and Future Works. This work only considers the planar state-based rearrangement with simplified environments where the agent can directly control the velocities of objects. In future works, it is necessary to extend our framework to more realistic scenarios, *e.g.*, manipulating objects by robots in a realistic 3D environment [11, 48]. We summarise the future directions in the following:

- *Visual Observation:* We may explore how to conduct efficient and accurate score-based reward estimation from image-based examples and leverage the recent progress on visual state representation learning [49, 50, 51, 52] to develop the visual gradient-based action.
- *Complex Dynamics:* When the agent can only move one object at a time, we may explore hierarchical methods where high-level policy chooses which object to move and low-level policy leverages the gradient-based action for motion planning. When we can only impose forces on objects instead of velocities, we may explore a hierarchical policy where a high-level policy outputs target velocities and a low-level employ a velocity controller, such as a PID controller, to mitigate the velocity errors.
- *Multi-Agent Scenarios:* The framework can also be extended to multi-agent scenarios [19, 53, 54, 55], where the decentralised agents need to estimate its gradient field and take actions according to the local observation for multi-agent cooperation or competition.

Ethics Statement. Our method has the potential to build home-assistant robots. We evaluate our method in simulated environments, which may introduce data bias. However, similar studies also have such general concerns. We do not see any possible major harm in our study.

Acknowledgement

This work was supported by the National Natural Science Foundation of China -Youth Science Fund (No.62006006): Learning Visual Prediction of Interactive Physical Scenes using Unlabelled Videos and China National Post- doctoral Program for Innovative Talents (Grant No. BX2021008). We also thank Yizhou Wang, Tianhao Wu and Yali Du for their insightful discussions.

References

[1] Yun Jiang, Marcus Lim, and Ashutosh Saxena. Learning object arrangements in 3d scenes using human context. In *ICML*, 2012.

[2] Nichola Abdo, Cyrill Stachniss, Luciano Spinello, and Wolfram Burgard. Robot, organize my shelves! tidying up objects by predicting user preferences. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1557–1564. IEEE, 2015.

[3] Henry N Ricciuti. Object grouping and selective ordering behavior in infants 12 to 24 months old. *Merrill-Palmer Quarterly of Behavior and Development*, 11(2):129–148, 1965.

[4] Ivan Kapelyukh and Edward Johns. My house, my rules: Learning tidying preferences with graph neural networks. In *5th Annual Conference on Robot Learning*, 2021.

[5] Hanqing Wang, Wei Liang, and Lap-Fai Yu. Scene mover: Automatic move planning for scene arrangement by deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

[6] Ju-Bong Kim, Ho-Bin Choi, Gyu-Young Hwang, Kwihoon Kim, Yong-Geun Hong, and Youn-Hee Han. Sortation control using multi-agent deep reinforcement learning in n-grid sortation system. *Sensors*, 20(12):3401, 2020.

[7] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020.

[8] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3357–3364. IEEE, 2017.

[9] Wenhan Luo, Peng Sun, Fangwei Zhong, Wei Liu, Tong Zhang, and Yizhou Wang. End-to-end active object tracking and its real-world deployment via reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(6):1317–1332, 2019.

[10] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[11] Yiming Zuo, Weichao Qiu, Lingxi Xie, Fangwei Zhong, Yizhou Wang, and Alan L Yuille. Craves: Controlling robotic arm with a vision-based economic system. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4214–4223, 2019.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[13] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems*, 29:4565–4573, 2016.

[14] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. {SQIL}: Imitation learning via reinforcement learning with sparse rewards. In *International Conference on Learning Representations*, 2020.

[15] Minghuan Liu, Tairan He, Minkai Xu, and Weinan Zhang. Energy-based imitation learning. *arXiv preprint arXiv:2004.09395*, 2020.

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

[17] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

[18] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

[19] Fangwei Zhong, Peng Sun, Wenhan Luo, Tingyun Yan, and Yizhou Wang. Towards distraction-robust active visual tracking. In *International Conference on Machine Learning*, pages 12782–12792. PMLR, 2021.

[20] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. *Advances in Neural Information Processing Systems*, 31, 2018.

[21] Ben Eysenbach, Sergey Levine, and Russ R Salakhutdinov. Replacing rewards with examples: Example-based policy search via recursive classification. *Advances in Neural Information Processing Systems*, 34:11541–11552, 2021.

[22] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[24] Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular conformation generation. In *International Conference on Machine Learning*, pages 9558–9568. PMLR, 2021.

[25] Shitong Luo and Wei Hu. Score-based point cloud denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4583–4592, 2021.

[26] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[27] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

[28] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[29] Jur van den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics Research*, pages 3–19. Springer, 2011.

[30] Martin J Schuster, Dominik Jain, Moritz Tenorth, and Michael Beetz. Learning organizational principles in human environments. In *2012 IEEE International Conference on Robotics and Automation*, pages 3867–3874. IEEE, 2012.

[31] Nichola Abdo, Cyrill Stachniss, Luciano Spinello, and Wolfram Burgard. Organizing objects by predicting user preferences through collaborative filtering. *The International Journal of Robotics Research*, 35(13):1587–1608, 2016.

[32] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3d object arrangements. *ACM Transactions on Graphics (TOG)*, 31(6):1–11, 2012.

[33] Lap Fai Yu, Sai Kit Yeung, Chi Keung Tang, Demetri Terzopoulos, Tony F Chan, and Stanley J Osher. Make it home: automatic optimization of furniture arrangement. *ACM Transactions on Graphics (TOG)-Proceedings of ACM SIGGRAPH 2011, v. 30,(4), July 2011, article no. 86*, 30(4), 2011.

[34] Yi-Ting Yeh, Lingfeng Yang, Matthew Watson, Noah D Goodman, and Pat Hanrahan. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Transactions on Graphics (TOG)*, 31(4):1–11, 2012.

[35] Ankur Handa, Viorica Pătrăucean, Simon Stent, and Roberto Cipolla. Scenenet: An annotated model generator for indoor scene understanding. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5737–5743. IEEE, 2016.

[36] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-centric indoor scene synthesis using stochastic grammar. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5899–5908, 2018.

[37] Dhruv Batra, Angel X Chang, Sonia Chernova, Andrew J Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, et al. Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv:2011.01975*, 2020.

[38] Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. Visual room rearrangement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5922–5931, 2021.

[39] Yash Kant, Arun Ramachandran, Sriram Yenamandra, Igor Gilitschenski, Dhruv Batra, Andrew Szot, and Harsh Agrawal. Housekeep: Tidying virtual households using commonsense reasoning. *arXiv preprint arXiv:2205.10712*, 2022.

[40] Gabriel Sarch, Zhaoyuan Fang, Adam W Harley, Paul Schydlo, Michael J Tarr, Saurabh Gupta, and Katerina Fragkiadaki. Tidee: Tidying up novel rooms using visuo-semantic commonsense priors. *arXiv preprint arXiv:2207.10761*, 2022.

[41] Aviv Netanyahu, Tianmin Shu, Joshua Tenenbaum, and Pulkit Agrawal. Discovering generalizable spatial goal representations via graph-based active reward learning. In *International Conference on Machine Learning*, pages 16480–16495. PMLR, 2022.

[42] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

[43] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations*, 2019.

[44] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[45] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.

[46] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021.

[47] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International Conference on Machine Learning*, pages 40–49. PMLR, 2018.

[48] Weichao Qiu, Fangwei Zhong, Yi Zhang, Siyuan Qiao, Zihao Xiao, Tae Soo Kim, Yizhou Wang, and Alan Yuille. Unrealcv: Virtual worlds for computer vision. In *ACM International Conference on Multimedia*, pages 1221–1224, 2017.

[49] Drew Linsley, Girik Malik, Junkyung Kim, Lakshmi Narasimhan Govindarajan, Ennio Mingolla, and Thomas Serre. Tracking without re-recognition in humans and machines. *Advances in Neural Information Processing Systems*, 34:19473–19486, 2021.

[50] Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019.

[51] Zhixuan Lin, Yi-Fu Wu, Skand Peri, Bofeng Fu, Jindong Jiang, and Sungjin Ahn. Improving generative imagination in object-centric world models. In *International Conference on Machine Learning*, pages 6140–6149. PMLR, 2020.

[52] Fangwei Zhong, Peng Sun, Wenhan Luo, Tingyun Yan, and Yizhou Wang. Ad-vat+: An asymmetric dueling mechanism for learning and understanding visual active tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1467–1482, 2019.

[53] Yuanfei Wang, Fangwei Zhong, Jing Xu, and Yizhou Wang. Tom2c: Target-oriented multi-agent communication and cooperation with theory of mind. In *International Conference on Learning Representations*, 2022.

[54] Jing Xu, Fangwei Zhong, and Yizhou Wang. Learning multi-agent coordination for enhancing target coverage in directional sensor networks. *Advances in Neural Information Processing Systems*, 33:10053–10064, 2020.

[55] Xuehai Pan, Mickel Liu, Fangwei Zhong, Yaodong Yang, Song-Chun Zhu, and Yizhou Wang. MATE: Benchmarking multi-agent reinforcement learning in distributed target coverage control. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[56] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne P. Tchapmi, Micael E. Tchapmi, Kent Vainio, Josiah Wong, Li Fei-Fei, and Silvio Savarese. igibson 1.0: a simulation environment for interactive tasks in large realistic scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page accepted. IEEE, 2021.

[57] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[58] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in Neural Information Processing Systems*, 33:12438–12448, 2020.

[59] Denis Yarats and Ilya Kostrikov. Soft actor-critic (sac) implementation in pytorch. https://github.com/denisyarats/pytorch_sac, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] see Sec. 1
   (b) Did you describe the limitations of your work? [Yes] see Sec. 7
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] see supplementary
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] see Sec. 7 and supplementary
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] please see the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] more details in the supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] more details in the supplemental material.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] more details in the supplemental material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Details of Rearrangement Tasks

A.1 Ball Rearrangement

The three tasks in ball rearrangement only differ in state representation, target distribution and pseudo-likelihood function:

Circling: The state is represented as a fully-connected graph where each node contains the two-dimensional position of each ball $\mathbf{s}^i = [x, y]$. The target examples are drawn from an explicit process with an intractable density function: We first uniformly sample a legal centre $o = [x_o, y_o]$ that allows all balls to place in a circle around this centre without overlap. Then we uniformly sample a radius r_o from the legal radius range based on the sampled centre. After sampling the centre and radius, we uniformly place all balls in a circle centred in o with radius r_o. The pseudo likelihood function is defined as $\mathbf{F}_{proxy}(\mathbf{s}) = \exp^{-(\sigma_\theta + \sigma_r)}$, where σ_θ and σ_r denote the standard deviation of the angle between two adjacent balls and the distances from each ball to the centre of gravity of all balls, respectively. Intuitively, if a set of balls are arranged into a circle, then the σ_r and σ_θ should be close to zero, achieving higher pseudo-likelihood.

Clustering: The state is represented as a full-connected graph where each node contains the two-dimensional position of each ball $\mathbf{s}^i = [x, y]$ and the one-dimensional category feature of each ball $c^i \in \{0, 1, 2\}$. We generate the target examples in two stages: First, we sample the positions of each ball from a *Gaussian Mixture Model(GMM)* $p_{GMM} : \mathbb{R}^{K*2} \to \mathbb{R}^+$ with two modes:

$$
\begin{aligned}
p_{GMM}(\mathbf{s}) = \frac{1}{2} & \prod_{1 \le i \le \frac{K}{3}} \mathcal{N}((0.18 \sin(\frac{0}{3}\pi), 0.18 \cos(\frac{0}{3}\pi)), 0.05I)(\mathbf{s}^i) \\
& \prod_{\frac{K}{3} \le i \le \frac{2K}{3}} \mathcal{N}((0.18 \sin(\frac{1}{3}\pi), 0.18 \cos(\frac{1}{3}\pi)), 0.05I)(\mathbf{s}^i) \\
& \prod_{\frac{2K}{3} \le i \le K} \mathcal{N}((0.18 \sin(\frac{2}{3}\pi), 0.18 \cos(\frac{2}{3}\pi)), 0.05I)(\mathbf{s}^i) \\
+ \frac{1}{2} & \prod_{1 \le i \le \frac{K}{3}} \mathcal{N}((0.18 \sin(\frac{0}{3}\pi), 0.18 \cos(\frac{0}{3}\pi)), 0.05I)(\mathbf{s}^i) \\
& \prod_{\frac{K}{3} \le i \le \frac{2K}{3}} \mathcal{N}((0.18 \sin(\frac{2}{3}\pi), 0.18 \cos(\frac{2}{3}\pi)), 0.05I)(\mathbf{s}^i) \\
& \prod_{\frac{2K}{3} \le i \le K} \mathcal{N}((0.18 \sin(\frac{1}{3}\pi), 0.18 \cos(\frac{1}{3}\pi)), 0.05I)(\mathbf{s}^i)
\end{aligned}
\tag{6}
$$

where $K = 21$ denotes the total number of balls. In the second stage, we slightly adjust the positions sampled from the GMM to eliminate overlaps between these positions by stepping the physical simulation. Since the results mainly depend on the GMM, we take the density function of GMM as the pseudo likelihood function $\mathbf{F}_{proxy}(\mathbf{s}) = p_{GMM}(\mathbf{s})$

Circling+Clustering: The state representation is the same as that of *Clustering*. To generate the target examples, we first generate a circle using the same process in *Circling*. Then, starting with any ball in the circle, we colour the ball three colours, in turn, $\frac{K}{3}$ for each. The order is randomly selected from red-yellow-blue and red-blue-yellow. This sampling is also explicit yet with an intractable density similar to *Circling*. The pseudo-likelihood function is defined as $\mathbf{F}_{proxy}(\mathbf{s}) = \exp^{-(\sigma_\theta + \sigma_r)} \cdot \exp^{-(\sigma_R + \sigma_G + \sigma_B) - \sigma_C}$ where σ_R denotes the standard deviation of the angle between two adjacent red balls, and σ_G and σ_B and σ_C denotes the standard deviation of the positions of red, green and blue centres. Intuitively, the first term $\exp^{-(\sigma_\theta + \sigma_r)}$ measures the pseudo likelihood of balls forming a circle and the next term $\exp^{-(\sigma_R + \sigma_G + \sigma_B) + \sigma_C}$ measures the pseudo likelihood of balls being clustered into three piles.

The common settings shared by each task are summarised as follows:

Horizon: Each training episode contains 100 steps.

Initial Distribution: We first uniformly sample rough locations for each ball, and then we eliminate overlaps between these positions by stepping physical simulation.

Dynamics: The floor and wall are all absolutely smooth planes. All the balls are bounded in an 0.3m x 0.3m area, with the radius being 0.025m. We set the friction coefficients of all balls to 100.0 since we observe that setting a small(*e.g.*not larger than 1.0) friction coefficient does not significantly affect the dynamics. Besides, to increase the complexity of the dynamics, we set the masses and restitution coefficients of all green and blue balls to 0.1 and 0.99, respectively. The masses and restitution coefficients of all red balls are set to 10 and 0.1, respectively. We observe that under these dynamics, the collision may significantly harm the efficiency of the rearrangement process. Hence, the agent has to adapt to the dynamics for more efficient object rearrangement.

Target Examples: We collect 100,000 examples for each task as target examples.

A.2 Room Rearrangement

Dataset and Simulator: We clean the 3D-Front dataset [46] to obtain *bedrooms* of rectangular shape and three to eight objects. We also drop the rooms with large objects or small free spaces. Since the number of different types of objects varies greatly, we drop the rooms with objects of rare category (*e.g.*, top cabinet). Each room is augmented by flipping two times and rotating four times to get eight variants. We then import these rooms into iGibson [56] to run the physical simulation. For a more efficient environment reset and physical simulation, we build a 'proxy simulator' based on PyBullet [57] to replace the original iGibson simulator. We use iGibson to load and save the metadata of each room. Then we reload these rooms in the proxy simulator, where each object is replaced by a simple box-shaped object with the same geometry.

The 756x8 rooms are used for *target examples*. The target examples are used for training the target score network, the classifier-based baselines and the VAE in goal-conditioned baselines. The other 83x8 rooms are used to initialise the room in the test phase: We first sample a room from the test split and then perform 1000 Brownian steps to obtain the initial state.

State and Action Spaces: The state consists of a aspect ratio $r_a \in \mathbb{R}^+$ and an object state $\mathbf{s}_o \in \mathbb{R}^{K\times 6}$ where K denotes the number of objects. The aspect ratio indicates the shape of the rectangular room $r_a = \tanh(\frac{b_x}{b_y})$ where b_x and b_y denotes the horizontal and vertical wall bounds. The object state is the concatenation of sub-states of all the objects $\mathbf{s}_o = [\mathbf{s}^1, \mathbf{s}^2, ...\mathbf{s}^i, ..., \mathbf{s}^K]$ where the sub-state of the i-th object $\mathbf{s}^i \in \mathbb{R}^6$ is consists of 2-D position, 1-D orientation, 2-D bounding box and an 1-D category label. The action is also a concatenation of sub-actions of all the objects $\mathbf{a}_o = [\mathbf{a}^1, \mathbf{a}^2, ...\mathbf{a}^i, ..., \mathbf{a}^K]$. For the i-th object, the action $\mathbf{a}^i \in \mathbb{R}^3$ consists of a 2-D linear and a 1-D angular velocity. The whole action space is normalised into a $3 \times K$ dimensional unit-box $[-1, 1]^{3\times K}$ by the velocity bounds.

Horizon: Each training episode contains 250 steps.

Initial Distribution: To guarantee the initial state is accessible to the high-density region of the target distribution, we sample an initial state in two stages: First, we sample a room from the 83x8 rooms in the test dataset. Then we perturb this room by 1000 Brownian steps.

Dynamics: We set the friction coefficient of all the objects in the room to zero, as the dynamics of the room are complex enough.

B Details of Our Method

B.1 Training the Target gradient Field

Complete training objective: The complete training objective is the SDE-based score-matching objective proposed by [23]:

$$\mathbb{E}_{t\sim\mathcal{U}(0,1)}\mathbb{E}_{\mathbf{s}(0)\sim p_{tar}(\mathbf{s})}\mathbb{E}_{\mathbf{s}(t)\sim p_{0t}(\mathbf{s}(t)|\mathbf{s}(0))}[\mathbf{\Phi}_{tar}(\mathbf{s}(t), t) - \nabla_{\mathbf{s}(t)} \log p_{0t}(\mathbf{s}(t) \mid \mathbf{s}(0))\|_2^2]. \quad (7)$$

where $p_{0t}(\mathbf{s}(t) \mid \mathbf{s}(0)) = \mathcal{N}(\mathbf{s}(t); \mathbf{s}(0), \frac{1}{2\log\sigma}(\sigma^{2t} - 1)\mathbf{I})$ and $\sigma = 25$ is a hyper-parameter.

Using this objective, we can obtain the estimated score *w.r.t.* different levels of the noise-perturbed target distribution $p^t_{tar}(\mathbf{s}(t)) = \int p_{0t}(\mathbf{s}(t) \mid \mathbf{s}(0))p_{tar}(\mathbf{s}(0))d\mathbf{s}(0)$ simultaneously. In this way, we

can efficiently try different noise-level for efficient hyperparameter tuning. The choice of t will be described in Sec. B.4.

Network Architecture: All the networks (*e.g.*, score network, actor and critic networks used in learning-based methods) used in our work are designed into three stages, *pre-processing stage*, *message passing stage* and *output stage*. The *pre-processing stage* aims at extract node feature for each object via linear layers so as to construct a fully connected graph for the next stage. The *message passing stage* takes the initial graph as input and passes them through several layers of graph convolutional layers. After the above two stages, *output stage* further encodes the feature of each node to obtain the node-wise output (*e.g.*, score, action). We denote the hidden dimension as d_h and embedding dimension as d_e. In room rearrangement, $d_h = 128, d_e = 64$ and in ball rearrangement $d_h = 64, d_e = 32$. For the other trivial details, we recommend looking up in our open-sourced codes.

We first encode the static feature $f_s^i \in \mathbb{R}^{d_e}$ and state feature $f_a^i \in \mathbb{R}^{d_h}$ for the i-th node with 2 linear layers. We further encode the noise feature $f_t \in \mathbb{R}^{d_e}$ by a Gaussian Fourier projection layer (used in [23]). For room rearrangement, we additionally encode the wall feature $f_w \in \mathbb{R}^{d_e}$ from the aspect ratio via two linear layers.

Then we construct a fully connected input graph where the content of the i-th node is the concatenation of static, state, noise and wall feature $[f_s^i, f_a^i, f_t, f_a] \in \mathbb{R}^{3*d_e+d_h}$ (for room rearrangement, $[f_s^i, f_a^i, f_t, f_a, f_w] \in \mathbb{R}^{4*d_e+d_h}$).

The input graph is passed through 3 (2 for room rearrangement) Edge Convolutional Layers where the inner network is two layers of MLPs with hidden size d_h. After the message passing, the 2 (3 for room rearrangement) dimensional node features serve as the score components on objects.

Following the parameterisation trick proposed by [58], we divide the score components by $\frac{1}{2\log\sigma}(\sigma^{2t} - 1)$.

B.2 Details of ORCA and Planning-based Framework

We set $\tau = 0.1$ and the simulation duration of each timestep $\Delta t = 0.02$. For each agent(object), ORCA only considers the 2-nearest agents as neighbours, since in our experiments ORCA often has no solution when the number of neighbours is larger than 2.

In all ball rearrangement tasks, we choose $t = 0.1$ as the initial noise scale for the target score network. The noise level decays to 0 within an episode.

B.3 Complete Derivation of Surrogate Objective

$$
\begin{aligned}
J(\pi) \iff & \mathbb{E}_{\rho(\mathbf{s}_0),\tau\sim\pi}[\sum_{\mathbf{s}_t\in\tau}\gamma^t \log p_{tar}(\mathbf{s}_t)] - \underbrace{\frac{\mathbb{E}_{\rho(\mathbf{s}_0)}[\log p_{tar}(\mathbf{s}_0)]}{1-\gamma}}_{constant} \\
=& \mathbb{E}_{\rho(\mathbf{s}_0),\tau\sim\pi}[\sum_{\mathbf{s}_t\in\tau}\gamma^t \log p_{tar}(\mathbf{s}_t)] - \mathbb{E}_{\rho(\mathbf{s}_0),\tau\sim\pi}[\sum_{\mathbf{s}_t\in\tau}\gamma^t \log p_{tar}(\mathbf{s}_0)] \\
=& \mathbb{E}_{\rho(\mathbf{s}_0),\tau\sim\pi}[\sum_{\mathbf{s}_t\in\tau}\gamma^t [\log p_{tar}(\mathbf{s}_t) - \log p_{tar}(\mathbf{s}_0)]] \\
=& \mathbb{E}_{\rho(\mathbf{s}_0),\tau\sim\pi}[\sum_{1\le t\le T}\gamma^t \sum_{1\le k\le t}[\log p_{tar}(\mathbf{s}_k) - \log p_{tar}(\mathbf{s}_{k-1})]] \overset{def}{=} J^*(\pi) \\
J^*(\pi) \approx& \mathbb{E}_{\rho(\mathbf{s}_0),\tau\sim\pi}[\sum_{1\le t\le T}\gamma^t \sum_{1\le k\le t}\langle \nabla_{\mathbf{s}} \log p(\mathbf{s}_{k-1}), \mathbf{s}_k - \mathbf{s}_{k-1}\rangle] \\
\approx& \mathbb{E}_{\rho(\mathbf{s}_0),\tau\sim\pi}[\sum_{1\le t\le T}\gamma^t \underbrace{\sum_{1\le k\le t}\langle \mathbf{\Phi}_{tar}(\mathbf{s}_{k-1}), \mathbf{s}_k - \mathbf{s}_{k-1}\rangle}_{r_{t-1}}] \overset{def}{=} \hat{J}(\pi)
\end{aligned}
\tag{8}
$$

B.4 Details of Learning-based Framework

Reward Function: For all the ball rearrangement tasks, we choose $t = 0.1$ for the target score network when outputting the gradient-based action $\mathbf{a}_t^g = \mathcal{G}(\mathbf{\Phi}_{tar}(\mathbf{s}_t, 0.1))$ and $t = 0.01$ when estimating the immediate reward $r_t = \langle \mathbf{\Phi}_{tar}(\mathbf{s}_t, 0.01), \mathbf{s}_{t+1} - \mathbf{s}_t \rangle$. At each time step t, each object also receives a collision penalty c_t^i. So the total reward for the i-th object at timestep t is $r_t^i = \langle \mathbf{\Phi}_{tar}(\mathbf{s}_t, 0.01), \mathbf{s}_{t+1} - \mathbf{s}_t \rangle + \lambda * c_t^i$ where λ denotes a hyper-parameter to balance the immediate reward and the collision penalty. We choose $\lambda = 5$ for *Clustering* and *Circling+Clustering*, $\lambda = 3$ for *Circling* and $\lambda = 0.2$ for room rearrangement. We conduct reward normalisation for both immediate reward and the collision penalty, which maintains a running mean μ_t and a standard deviationσ_t of a given reward sequence and returns a z-score $z_t = \frac{r_t - \mu_t}{\sigma_t}$ as the normalised reward. For room rearrangement, we choose $t = 0.01$ for both outputting the gradient-based action and estimating the immediate reward.

RL Backbone: We use Soft-Actor-Critic (SAC) [45] as our RL backbone and implement SAC based on an open-sourced PyTorch implementation on GitHub [59] with 300+ stars. We keep all the hyperparameters the same except that $\gamma = 0.95$ since the reward signal is dense in our case. We set training iteration to 500,000 for ball rearrangement and 1,000,000 for room rearrangement.

Actor Network: Similar to the target score network, we first encode the state feature $f_a^i \in \mathbb{R}^{d_h}$ and static feature $f_s^i \in \mathbb{R}^{d_e}$ for the i-th agent. We also compute the target gradient on the state $\mathbf{g} = \mathbf{\Phi}_{tar}(\mathbf{s}, t)$. For room rearrangement, we also additionally encode the wall feature $f_w \in \mathbb{R}^{d_e}$ from the aspect ratio via two linear layers.

The content of the i-th node of the input graph is the concatenation of the state feature, static feature and gradient component on the i-th object $[f_s^i, f_a^i, \mathbf{g}_i] \in \mathbb{R}^{d_e + d_h + 3}$ (For room rearrangement $[f_s^i, f_a^i, f_w, \mathbf{g}_i] \in \mathbb{R}^{d_e * 2 + d_h + 3}$). After 1 (2 for room rearrangement) layers of message passing via Edge Convolution where the inner network is two layers of MLPs with hidden size d_h, the 2×2 (3×2 for room rearrangement) dimensional node features serve as the mean and variance of the action distribution of the objects.

Critic Network: The architecture of the critic network is similar to the actor network. We additionally encode the action feature $f_{ac}^i \in \mathbb{R}^{d_e}$ from the i-th object's action component and then concatenate the action feature with the graph node content. After the message passing, each node contains a one-dimensional output. We mean pooling the outputs across all nodes to get the output of the critic network.

C Details of Baselines

Here we briefly describe the implementation details of baselines. We recommend directly searching for more details in the supplementary codes.

C.1 Goal-based Baselines

These baselines refer to the *Goal-SAC* and *Goal-ORCA* in experiments.

Goal Proposal: This type of baseline first train a VAE on the target examples and then leverages the trained VAE for the goal proposal. The VAE is implemented as a GNN, and the model capacity is similar to our target score network for a fair comparison. We choose $\lambda_{kl} = 0.01$ for *Circling* and *Clustering* and $\lambda_{kl} = 0.02$ for *Circling+Clustering*.

Execution: At the beginning of each episode, the agent first proposes a goal for this episode using the VAE and then reaches the goal via a control algorithm. In *Goal-ORCA*, the agent reaches the goal by a planning-based method: The agent first assigns velocities for each ball that points to the corresponding goal. Then these velocities are updated by the ORCA planner ϕ to be collision-free. In *Goal-SAC*, the agent trains a multi-agent goal-conditioned policy via goal-conditioned RL to reach the goal: Similar to *Ours-SAC*, the reward of the i-th object at timestep t is $r_t^i = ||\mathbf{s}_t - \mathbf{s}_{goal}||_1 + \lambda * c_t^i$ where $\mathbf{s}_{goal}$ denotes the goal proposal, λ is a hyper-parameter and $c_t^i = \sum_{j \neq i} col_{i,j}$ is the total number of collisions between the i-th object and the others. Here $col_{i,j} = 1$ when We choose $\lambda = 3$ for ball rearrangement and $\lambda = 0.2$ for room rearrangement.

C.2 Classifier-based Baselines

These baselines refer to the *RCE*, *SQIL* and *GAIL* in experiments.

RCE and *SQIL* are implemented based on the codes [21] released by RCE's authors. We only modify $\gamma = 0.95$, the training steps decrease to 0.5 million for ball rearrangement (*i.e.*, the same number of training steps as other methods) and the model architecture. The architecture of actor and critic networks is implemented the same as ours(*i.e.*, the same feature extraction layers and Edge Convolutional layers, except for the target gradient feature).

GAIL is actually a modification of our learning-based framework: Keeping the RL agent the same(*i.e.*, multi-agent SAC), *GAIL*'s reward is given by a discriminator. The architecture of the discriminator is the same as our critic network, except that the input graph does not contain the action feature.

At each training step, we update the discriminator by distinguishing between the agents' and the expert's states(for one step) and then update the RL policy under the reward given by the discriminator(for one step). The agent also receives a collision reward during training similar to *Ours-SAC* and *Goal-SAC*: $r_t^i = D(\mathbf{s}_{t+1}) + \lambda * c_t^i$ where D denotes the classifier trained by *GAIL*. We do not conduct reward normalisation for *GAIL* as the learned reward is unstable. We choose $\lambda = 10$ for room rearrangement, *Circling* and *Circling+Clustering*, $\lambda = 100$ for *Clustering*.

D Details of Evaluations

D.1 Ball rearrangement

We collect 100 trajectories for each task starting from the same set of initial states. To calculate the coverage score, we sample fixed sets examples from the target distribution serving as S_{gt} for *Circling*, *Clustering*, and *Circling+Clustering*, respectively. We sample 20 examples for *Circling* and *Circling+Clustering* and 50 examples for *Clustering*. Since the balls in the same category can actually be viewed as a two-dimensional point cloud, we measure the distance between two states by summing the CDs between each pile of balls by category.

D.2 Room rearrangement

For each room in 83 test rooms, we collect eight initial states. Then we collect 83x8 trajectories starting from these initial states for each method. The coverage score is calculated by averaging the coverage score in each room condition since the state dimension differs in different rooms. For each room in 83 test rooms, we calculate the coverage score between the eight ground truth states and eight rearrangement results and then the averaged coverage score over the 83 rooms is taken as the final coverage score for a method. We measure the distance between two states by calculating the average L2 distance between the positions (*i.e.*, we ignore the orientations) of the corresponding objects.

E Additional Results

E.1 Single-mode Problem of Ours w/o Residual

In Fig. 7 we show qualitative results of *Ours w/o Residual*. Apparently, the balls are arranged into a single pattern in all three ball rearrangement tasks, while the examples from the target distribution are diverse. In the most difficult task *Circling + Clustering*, the agent cannot even reach a terminal state with a high likelihood. This result indicates that *Ours w/o Residual* failed to explore the high-density region of target distribution without residual learning.

E.2 Effectiveness of Reward Learning

In each ball rearrangement task, we collect 100 trajectories, each of which is run by a hybrid policy. The hybrid policy takes the first 50 steps using the *Ours (ORCA)* and the next 50 steps using random actions. To evaluate the effectiveness of our method on reward learning, we compare the estimated reward curve of *Ours (SAC)* and *GAIL* with the *pseudo likelihood(PL)* curve of the trajectories.

As shown in 8, the reward curve of *Ours (SAC)* best fits the trend of the pseudo-likelihood curve, which shows the effectiveness of our reward estimation method.

E.3 Efficiency Problem of Goal-conditioned Baselines

In Fig. 1, we report the comparative results of our framework and goal-based baselines(*e.g.*, *Goal (SAC)*, *Goal (ORCA)*) on a new metric named *absolute state change(ASC)*. The ASC measures the sum of the absolute paths of all small balls in the rearrangement process.

$$\text{ASC} = \sum_{1 \leq t \leq T} \sum_{1 \leq k \leq K} ||\mathbf{s}_t^k - \mathbf{s}_{t-1}^k||_1 \tag{9}$$

As shown in Fig. 1, *Ours (SAC)* and *Ours (ORCA)* are significantly better than *Goal (SAC)* and *Goal (ORCA)*, respectively. This result explains why our method's likelihood curves are better than the goal-based baselines': The proposed goal is far away from the initial state, which harms the efficiency of goal-based approaches.

Table 1: Quantitative comparison results of our framework(*Ours (ORCA)*, *Ours (SAC)*) and goal-based baselines(*Goal (ORCA)*, *Goal (SAC)*) in rearrangement efficiency. For each method, we report the mean and standard deviation of *absolute state change* over 100 episodes on each ball rearrangement task.

Method	Circling		Clustering		Circling+Clustering	
	21 balls	30 balls	21 balls	30 balls	21 balls	30 balls
Ours-ORCA	**23.16 +- 2.42**	**29.08 +- 2.43**	**13.72 +- 1.40**	**16.60 +- 1.68**	**19.54 +- 2.19**	**23.29 +- 2.18**
Goal-ORCA	28.05 +- 2.82	31.98 +- 2.90	27.57 +- 2.90	33.37 +- 4.06	27.77 +- 2.86	32.47 +- 2.99
Ours-SAC	**56.76 +- 4.83**	**78.18 +- 6.75**	**65.35 +- 4.64**	**110.06 +- 4.25**	**48.93 +- 4.68**	**80.01 +- 6.19**
Goal-SAC	108.25 +- 7.53	139.26 +- 8.24	118.15 +- 5.84	142.05 +- 7.56	122.72 +- 5.93	161.01 +- 6.84

E.4 Visualisations of Goals Proposed by the VAE

We demonstrate the visualisations of goal proposals by the VAE used in goal-based baselines. Typically, the proposed goals indeed form a reasonable shape that is similar to the target examples. However, it is hard to generate a fully legal goal since the VAE is not accessible to the dynamics of the environment or has enough data to infer the physical constraints of the environment. As shown in Fig. 9, there exist many overlaps between balls in generated goals, which causes the balls to have conflicting goals and thus harms the efficiency of goal-based baselines.

E.5 Six-modes Clustering

Task Settings: This task is a six-modes extension of *Clustering*, where the centres of clusters are located in the following six patterns.

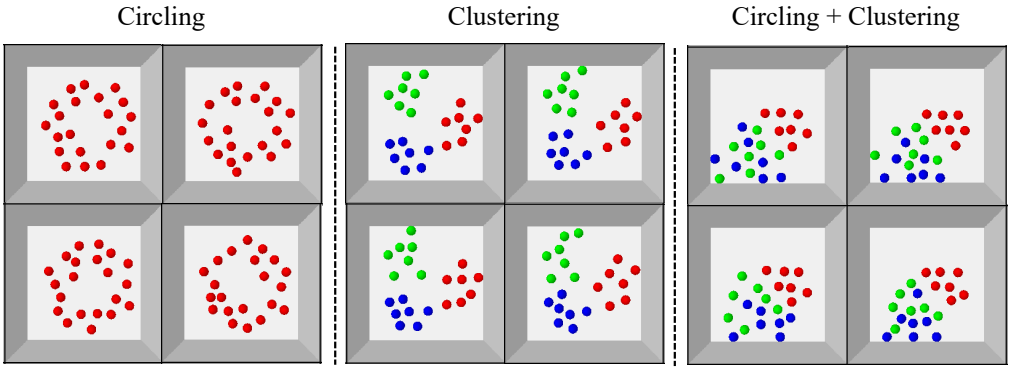


Figure 7: We visualise rearrangement results of *Ours w/o Residual* to demonstrate the 'single pattern' phenomenon.

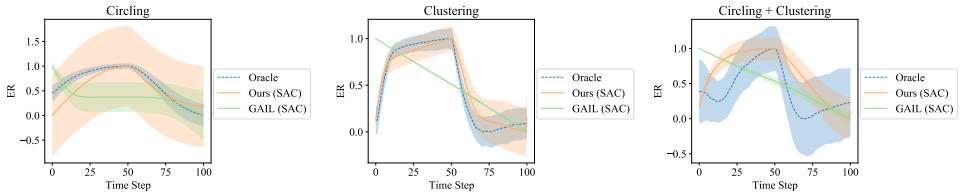


Figure 8: Given a set of identical trajectories, we compare the *estimated reward(ER)* of different methods and *pseudo likelihood(PL)*. All curves are normalised to the range of [- 1,1].

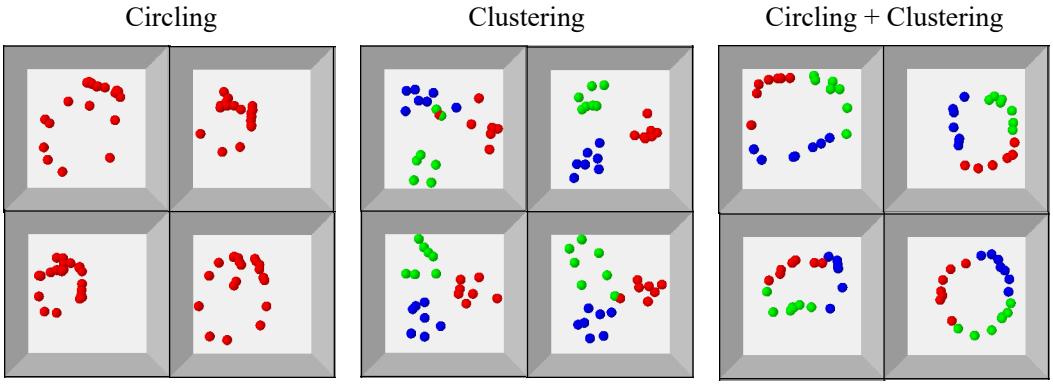


Figure 9: We visualise the goals proposed by the VAE used in *Goal (ORCA)* and *Goal (SAC)*.

Defining the joint centres' positions as a latent variable $C = (C_r, C_g, C_b)$ where C_r, C_g and C_b denotes centres of red, green and blue balls respectively and the above six modes as $\{c_i\}_{1\leq i\leq 6}$, the C obeys a categorical distribution $p(C = c_i) = \frac{1}{6}$.

The target distribution of six-modes clustering is a Gaussian Mixture Model:

$$p_{GMM}(\mathbf{s}) = \sum_{1\leq k\leq 6} p(C = c_k)p(\mathbf{s}|C = c_k)$$
$$p(\mathbf{s}|C = c_k) = \prod_{1\leq i\leq \frac{K}{3}} \mathcal{N}(C_r^k, 0.05I)(\mathbf{s}^i) \prod_{\frac{K}{3}\leq i\leq \frac{2K}{3}} \mathcal{N}(C_g^k, 0.05I)(\mathbf{s}^i) \prod_{\frac{2K}{3}\leq i\leq K} \mathcal{N}(C_b^k, 0.05I)(\mathbf{s}^i) \tag{10}$$

Notably, the 'mean mode' of above six modes is the origin, *i.e.*, $\frac{1}{6}\sum_{i=1}^{6} c_i = ((0,0),(0,0),(0,0))$. If the policy arranges the balls into this mean pattern, then the balls should be centred around $(0,0)$ (*i.e.*, all positions of balls obey $\mathcal{N}(0, 0.05I)$). As shown in Fig. 10 (a), we illustrate an example for each mode.

Implementation Details: We evaluate Ours (SAC) and Ours (ORCA) on this task. The performances are normalised by the Oracle.

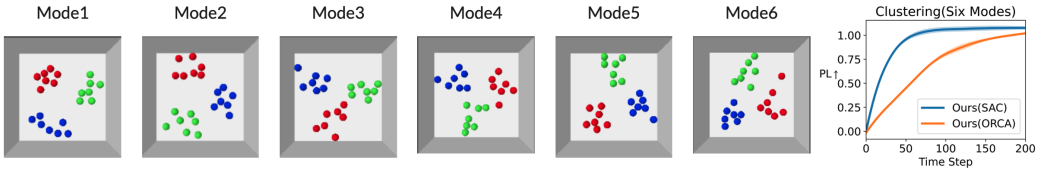
Results: We report the pseudo-likelihood curves of our methods. As shown in Fig. 10 (b), the rearrangement results are close to ground truth examples from the target distribution.

We further compute the latent distributions for rearrangement results of Ours (SAC) using the Bayesian Theorem:

$$p(C = c_i|\mathbf{s}) = \frac{p(\mathbf{s}|C = c_i)p(C = c_i)}{\sum_j p(\mathbf{s}|C = c_j)p(C = c_j)} \tag{11}$$

$p(C = c_i|\mathbf{s})$ indicates which mode a state belongs to. To demonstrate our rearrangement results are close to one of the six modes instead of concentrating on a 'mean' pattern, we evaluate the average entropy of the latent distributions $\mathbb{E}[H(p(C|\mathbf{s}))]$. As shown in Fig. E.5, the average entropy of our

Locations	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6
$(0.18\cos(\frac{2\pi}{3}), 0.18\sin(\frac{2\pi}{3}))$	R	R	B	B	G	G
$(0.18\cos(\frac{4\pi}{3}), 0.18\sin(\frac{4\pi}{3}))$	G	B	G	R	B	R
$(0.18\cos(2\pi), 0.18\sin(2\pi))$	B	G	R	G	R	B



(a) (b)

Figure 10: Pseudo likelihood curves and qualitative results of six-modes clustering.

methods is even lower than ground truths'. This indicates each state in our rearrangement results distinctly belonging to one of the categories.

We also report the average latent distribution $\mathbb{E}[p(C|\mathbf{s})]$. As shown in Fig. E.5, the averaged latent distribution (overall rearrangement results) of our methods achieve comparable orders of magnitude in different modes. This shows the rearrangement of our methods can cover all the mode centres.

E.6 Move One-ball at a Time

Task Settings: This task is an extension of *Clustering* where the policy can only move one ball at a time. We increase the horizon of each episode from 100 to 300. At each time step, the agent can choose one ball to take a velocity-based action for 0.1 seconds.

Implementation Details: We design a bi-level approach based on our method as shown in Fig. 11 (b): Every 20-time steps, the high-level planner outputs an object index i_t with the largest target gradient's component:

$$i_t = \underset{i}{\operatorname{argmax}} ||\mathbf{g}_t^i||_2 \tag{12}$$

where $\mathbf{g}_t^i \in \mathbb{R}^2$ denotes the component of the target gradient on the i-th object. In the following 20 steps, the ORCA planner computes the target velocity according to $\mathbf{g}_t^i$ and masks all the other object's velocities to zero.

We compare our method with a goal-based baseline where the agent first generates goals for each object via the VAE used in Goal (ORCA). Then the high-level planner chooses the object with the farthest distance to the goal, denoted as i_t. The low-level planner of this baseline is the same as ours.

Results: As shown in Fig. 11 (a), (c), our method achieves more appealing results, better efficiency and performance compared with the goal-based baseline.

E.7 Force-based Dynamics

Task Settings: This task is an extension of *Circling + Clustering* where the agent can only impose forces on the objects instead of velocities. At each time step, the agent can assign a two-dimensional force $f_i \in \mathbb{R}^2$ on each object.

Implementation Details: Similar to the 'one at a time' experiment, we design a bi-level approach to tackle this task, as Fig. 12 (b) illustrates. The high-level policy outputs a target velocity every eight steps. In the following eight steps, after the target velocities are outputted, the low-level PID

Methods	Average Entropy ↓	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6
GT	0.0066	0.17	0.17	0.17	0.17	0.17	0.17
Ours(SAC)	0.0037	0.15	0.18	0.31	0.14	0.10	0.13
Ours(ORCA)	0.0056	0.17	0.17	0.16	0.11	0.19	0.19

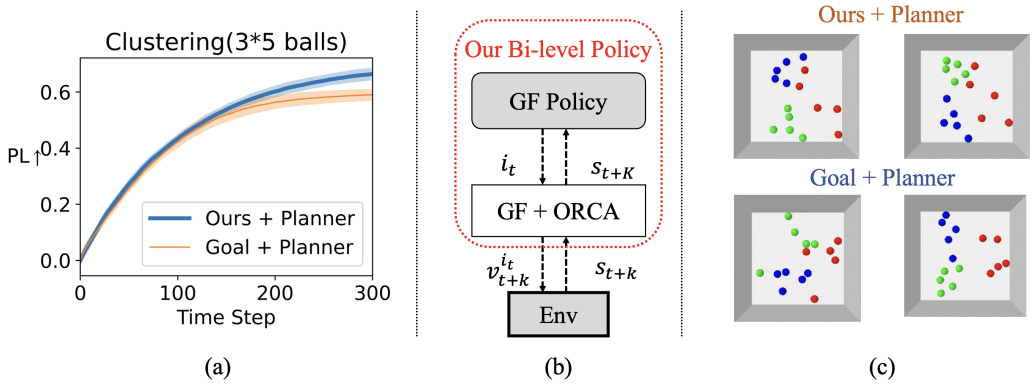


Figure 11: (a): Quantitative results. (b): Illustration of our bi-level policy. (c): Qualitative results.

controller receives the target velocity and outputs the force-based action to minimise the velocity error. We set $K_P = 10.0, K_I = 0.0, K_D = 0.0$ for PID controller.

This policy is compared with Ours (SAC) in the main paper.

Results: As shown in Fig. 12. (a) and (c), this force-based policy achieves comparable performance with Ours (SAC) yet suffers from a slight efficiency drop due to the control error of PID.

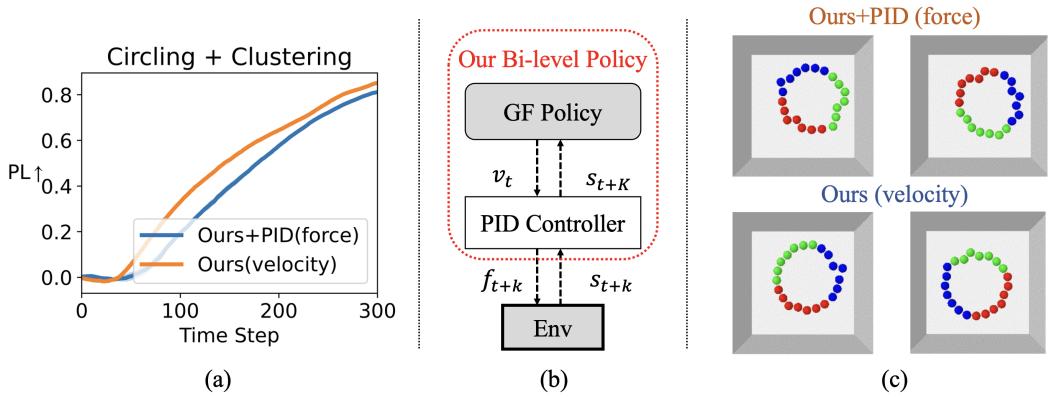


Figure 12: (a): Quantitative results. (b): Illustration of our bi-level policy. (c): Qualitative results.

E.8 Image-based Reward Learning

Task Settings: The target score network is trained on image-based target examples. We simply render the state-based target example set used in Ours (SAC) to 64x64x3 images and

Implementation Details: Our target score network of Ours(Image) is trained on the image-based examples set. The policy network and gradient-based action are state-based since we focus on image-based reward learning instead of visual-based policy learning.

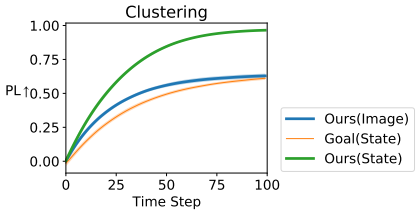


Figure 13: Quantitative results of image-based reward learning experiment.

This approach is compared with Ous(SAC) and Goal (SAC) in the main paper.

Results: Results in Fig. 13 show that Ours(Image) achieves slightly better performance than Goal (SAC) yet is lower than Ours (SAC) due to the increment of the dimension. This indicates that our reward learning method is still effective in image-based settings.