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 a radius of 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. All red balls' masses and restitution coefficients 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 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 $\mathbf{s}^i \in \mathbb{R}^6$ consists of 2-D position, 1-D orientation, 2-D bounding box and a 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 room's dynamics 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. This way, we

can efficiently try different noise levels 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 extracting node features for each object via linear layers 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 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 the embedding dimension as d_e. In room rearrangement, $d_h = 128, d_e = 64$ and in ball rearrangement $d_h = 64, d_e = 32$. We recommend looking up the other trivial details 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 we observe that 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 linearly 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 $r_t = \langle \mathbf{\Phi}_{tar}(\mathbf{s}_t, 0.01), \mathbf{s}_{t+1} - \mathbf{s}_t \rangle$. Our experiments found that the choice of $t = 0.01$ for the gradient-based action works better. For a fair comparison with Ours(ORCA), we still set $t = 0.1$ for the gradient-based action. 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 $t = 0.01$ to output the gradient-based action and estimate 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.

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

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.

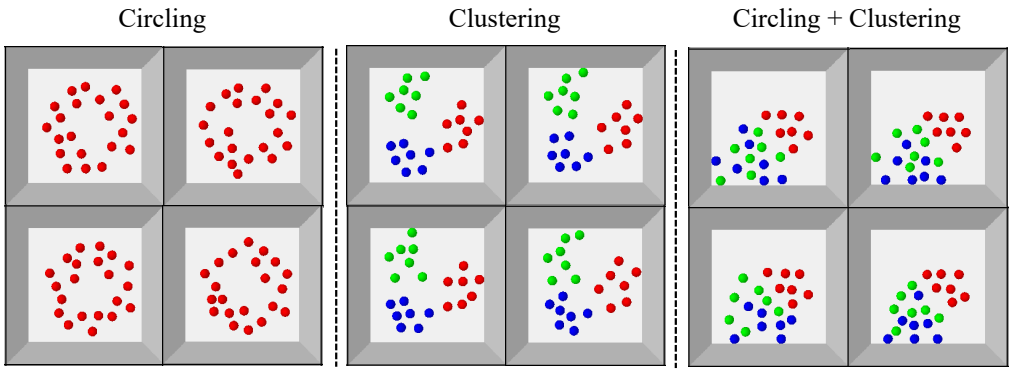


Figure 7: We visualise rearrangement results of *Ours w/o Residual* to demonstrate the 'single pattern' phenomenon.

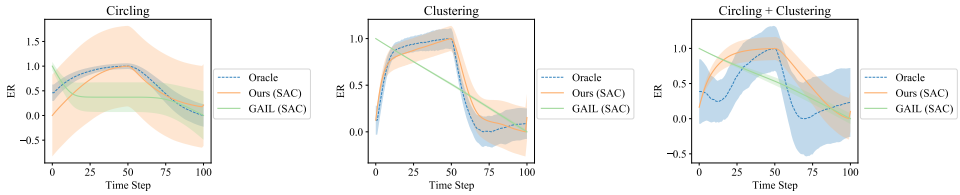


Figure 8: Given a set of identical trajectories, we compare the *estimated reward(ER)* of different methods and *pseudo likelihood(PL)*. All curves are normalised to the range of [- 1,1].

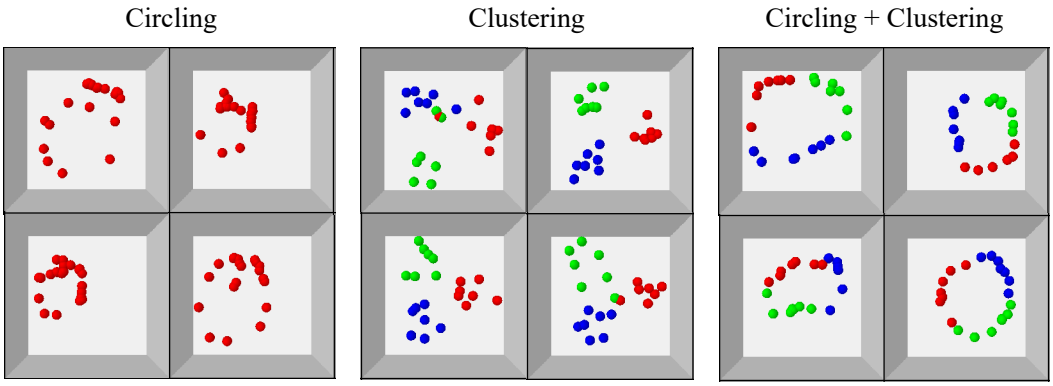


Figure 9: We visualise the goals proposed by the VAE used in *Goal (ORCA)* and *Goal (SAC)*.

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.

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

Defining the joint centres' positions as a latent variable $C = (C_r, C_g, C_b)$ where C_r, C_g and C_b denote 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:

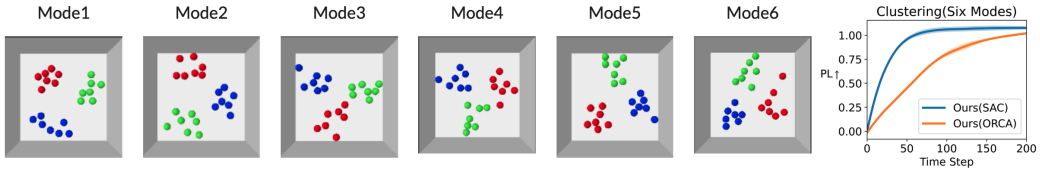
$$
\begin{aligned}
p_{GMM}(\mathbf{s}) &= \sum_{1\leq k\leq 6} p(C = c_k)p(\mathbf{s}|C = c_k) \\
p(\mathbf{s}|C = c_k) &= \prod_{1\leq i\leq \frac{K}{3}} \mathcal{N}(C_r^k, 0.05I)(\mathbf{s}^i) \prod_{\frac{K}{3}\leq i\leq \frac{2K}{3}} \mathcal{N}(C_g^k, 0.05I)(\mathbf{s}^i) \prod_{\frac{2K}{3}\leq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

Notably, the 'mean mode' of the 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.

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

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.



(a) (b)

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

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 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, our methods' averaged latent distribution (overall rearrangement results) 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:

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

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 other objects' velocities to zero.

We compare our method with a goal-based baseline where the agent 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.

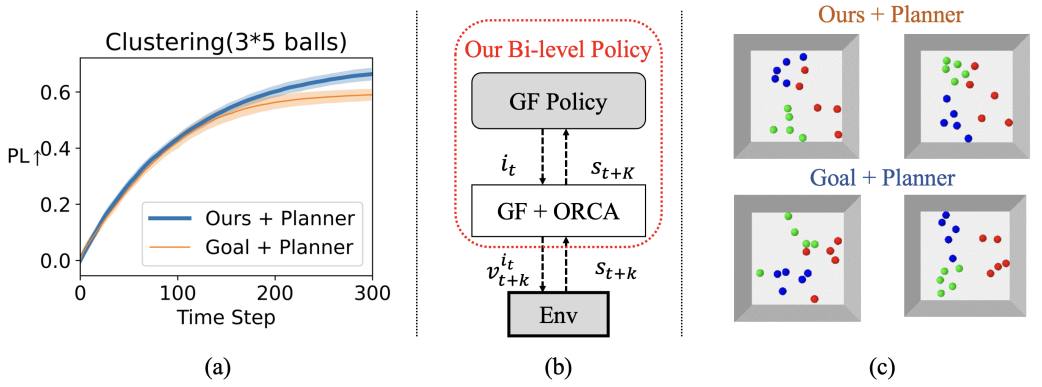


Figure 11: (a): Quantitative results. (b): Illustration of our bi-level policy. (c): Qualitative results.

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 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.

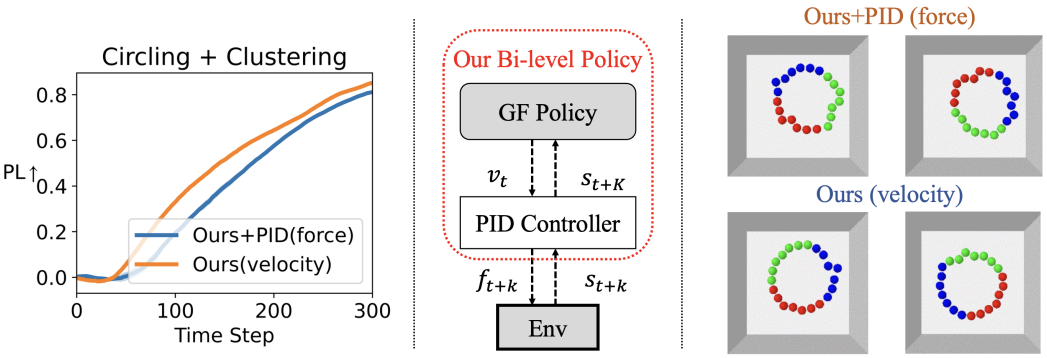


Figure 12: (a): Quantitative results. (b): Illustration of our bi-level policy. (c): Qualitative results.

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-

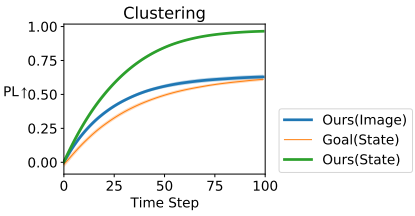


Figure 13: Quantitative results of image-based reward learning experiment.

based since we focus on image-based reward learning instead of visual-based policy learning.

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.