

A Lightweight behavior cloning and online recovery analysis

We further analyze the interplay between lightweight behavior cloning and online recovery. On the random1 layout, we start with a pretrained agent and conduct 1-epoch behavior cloning on the “liked” trajectories for each of the three preference styles. This is followed by online recovery using the policy pool. The corresponding training curves are shown in Figure 4.



Figure 4: Training curve of online recovery after one epoch of preference behavior cloning. The blue horizontal line indicates the environment return of the pretrained agent on the policy pool. The yellow curve shows the agent’s action accuracy on the offline “liked” trajectories during recovery.

Our results reveal that even a minimal application of offline preference learning—namely, a single round of behavior cloning on a small dataset—can lead to a substantial degradation in generalization performance, with environment returns dropping by more than one-third. This highlights a clear inefficiency mismatch between behavior cloning and recovery, where the latter requires thousands of online interaction episodes to restore the agent’s original performance level.

Interestingly, we observe that after recovery, the agent not only regains its generalization ability but also retains partial alignment with the offline “liked” trajectories. This indicates that preference information is not entirely forgotten during recovery, providing empirical support for the effectiveness of our Epoch-wise Alternation Recovery in reinforcing preference learning while maintaining generalization.

B Overcooked benchmark details

The Overcooked environment[Carroll et al., 2019] is a widely adopted benchmark for evaluating cooperative behaviors in reinforcement learning. Inspired by the multiplayer video game Overcooked!, it simulates a kitchen where two players, each controlling a chef, must coordinate within a confined space and under time constraints to prepare and deliver soups. Upon successful delivery, both players receive a shared reward. The primary objective is to maximize the cumulative team reward through effective collaboration.

To prepare a soup, agents typically follow a multi-step process: (1) collect and place the correct ingredients into a pot according to the recipe; (2) wait for the soup to cook for a fixed duration; and (3) serve the cooked soup on a dish and deliver it to the designated location. Each agent operates in a discrete action space: up, down, left, right, interact, no-op. Beyond learning the mechanics of soup preparation, agents must also adapt to the behavioral tendencies and preferences of their partners. High team performance requires precise coordination and mutual understanding between agents.

B.1 Layouts

Our work builds upon the Overcooked version used in the HSP benchmark[Yu et al., 2023], primarily focusing on two layouts: Coordination Ring and Many Orders.

As shown in the left part of Figure 5, Coordination Ring features a ring-shaped kitchen layout. The tightly arranged space facilitates faster soup preparation, with ingredients, serving station, and plates all located in the bottom-left corner, and two pots positioned in the top-right. The key coordination challenge in this layout lies in the agents’ movement directions to avoid collisions. The recipe is



Figure 5: Two layouts in our experiments.

517 relatively simple—only onion soup is required—and each soup takes 20 time steps to cook, yielding
 518 a reward of 20.

519 Many Orders, in contrast, adopts a square-shaped kitchen and introduces greater recipe complexity.
 520 It includes three types of orders: onion, tomato, and a mixed recipe requiring 1 onion and 2 tomatoes.
 521 Each recipe provides a different reward. In this layout, the central coordination challenge is recipe
 522 selection. If the human partner shows a preference for a particular recipe, the cooperative agent is
 523 expected to align with that preference to achieve satisfactory collaboration.

524 B.2 Events

525 Following the settings established in HSP, we adopt the same event taxonomy to define key sub-events
 526 within each layout. For both layouts (Coordination Ring and Many Orders), the primary sub-events
 527 and their corresponding event scores are listed as following Table 4,5 :

Table 4: Event categories and responding reward weights of Layout Coordination Ring

Event (e_i)	Event reward weight ($\text{score}(e_i)$)
Picking up an onion from the onion dispenser	-10, 0, 10
Picking up a dish from the dish dispenser	0, 10
Picking up a ready soup from the pot with a dish	-10, 0, 10
Placing an onion into the pot	-10, 0, 10
Delivering a soup to the serving area	-10, 0

Table 5: Event categories and responding reward weights of Layout Many Orders

Event (e_i)	Event reward weight ($\text{score}(e_i)$)
Picking up an onion from the onion dispenser	-10, 0, 10
Picking up a dish from the dish dispenser	0, 10
Picking up a tomato from tomato dispenser	0, 10, 20
Picking up a soup	-5, 0, 5
Viable placement	-10, 0, 10
Optimal placement	-10, 0
Catastrophic placement	0, 10
Placing an onion into the pot	-3, 0, 3
Placing a tomato into the pot	-3, 0, 3
Delivering a soup to the serving area	-10, 0

528 B.3 Offline "like" trajectories

529 For the offline datasets, we collect trajectories across two layouts and three preference styles, with
 530 the number of trajectories for each combination summarized in Table 6. For each layout-style pair,

we split the trajectories into training and test sets using a 4:1 ratio. In our reported results, action accuracy refers to performance evaluated on the test set.

Table 6: Number of trajectories in the offline datasets across two layouts and three preference styles. Each set is split into training and test sets with a 4:1 ratio.

Layout	Style A	Style B	Style C	total
Coordination Ring	12	24	20	72
Many Orders	38	14	12	72

C Hyper parameters

In this section, we present the hyperparameters used and compute resources in our experiments.

C.1 Agent architecture

Following the experimental settings in HSP, we adopt a CNN-MLP based architecture for all experiments. The convolutional module consists of three layers with output channels {32, 64, 32}. Each layer uses a kernel size of 3, padding of 1, and stride of 1. A max pooling layer is applied after the convolutional block, followed by a two-layer MLP that outputs the final action logits.

C.2 FCP policy pool

To construct the policy pool, we perform self-play training with the objective of maximizing environment rewards. A total of 6 self-play runs are conducted, and for each run, we select three checkpoints—initial, middle, and final—resulting in a policy pool containing 18 diverse agents. The hyperparameters used for the self-play procedure are listed in Table 7 :

Table 7: Hyperparameters of self-play process

Hyperparameters	Values
Entropy coef	0.01
Gradient clip norm	10.0
GAE lambda	0.95
Gamma	0.99
Value loss	huber loss
Huber delta	10.0
Mini batch size	batch size / mini-batch
Optimizer	Adam
Optimizer epsilon	1e-5
Weight decay	0
Network initialization	Orthogonal
Use reward normalization	True
Use feature normalization	True
Learning rate	5e-4
Parallel environment threads	100
Ppo epoch	15
Environment steps	10M
Episode length	400
Reward shaping horizon	100M

545 C.3 Pretraining, recovery, behavior cloning

546 Since pretraining and online generalization recovery share the same underlying procedure, we adopt
 547 identical hyperparameters for both stages. Additionally, the hyperparameters used for behavior
 548 cloning during preference learning are also summarized in Table 8.

Table 8: Hyperparameters of pretraining, online recovery, behavior cloning.

Hyperparameters	Values
BC batch size	32
BC epoch at stage 2	16
Hyperparameters	Values
N training threads	1
N rollout threads	200
Entropy coef	0.01
Gradient clip norm	10.0
GAE lambda	0.95
Gamma	0.99
Value loss	huber loss
Huber delta	10.0
Mini batch size	batch size / mini-batch
Optimizer	Adam
Optimizer epsilon	1e-5
Weight decay	0
Network initialization	Orthogonal
Use reward normalization	True
Use feature normalization	True
Learning rate	5e-4
Parallel environment threads	100
Ppo epoch	15
Environment steps	20M
Episode length	400
Reward shaping horizon	100M

549 D Results on the Many Orders Layout

Table 9: Performance across different preference styles. Each method is evaluated using Preference Score (\uparrow) and Environment Return (\uparrow). Results are reported on the layout *Many Orders*

Method	Style A		Style B		Style C	
	Pref. Score	Env. Return	Pref. Score	Env. Return	Pref. Score	Env. Return
Offline Trajs	244.7	203.4	669.3	336.4	1002.5	378.3
Pretrain	194.3 \pm 18.0	166.7 \pm 16.0	124.2 \pm 17.4	125.0 \pm 29.7	227.3 \pm 92.1	120.2 \pm 63.0
Pretrain + BC	145.0 \pm 16.8	116.4 \pm 13.8	334.0 \pm 4.5	340.0 \pm 2.2	616.3 \pm 27.1	337.3 \pm 15.2
BC Only	86.3 \pm 51.9	62.8 \pm 41.5	314.0 \pm 9.9	316.3 \pm 11.0	661.3 \pm 13.5	363.0 \pm 8.0
Pretrain + BC + Recover	213.7 \pm 5.7	178.7 \pm 4.6	228.3 \pm 44.1	230.3 \pm 59.9	390.7 \pm 102.9	217.0 \pm 54.3
EAR	212.7 \pm 4.5	178.3 \pm 4.2	239.0 \pm 53.8	239.3 \pm 68.8	395.3 \pm 117.1	222.3 \pm 65.8

Table 10: Average environment returns (\uparrow) across all benchmark human proxy agents π_h^i under different methods and preference styles in layout *Many Orders*. Since the pretrained agent is style-agnostic, its performance is reported only once.

Preference	BC	Pretrain + BC	EAR
Style A	33.6 ± 21.6	14.6 ± 3.6	146.3 ± 7.9
Style B	68.2 ± 4.5	86.5 ± 3.5	172.3 ± 26.0
Style C	65.0 ± 2.9	67.9 ± 2.6	195.3 ± 21.1
Pretrain	142.0 ± 22.9		

In this section, we present the experimental results on the Many Orders layout. All results are averaged over three runs with different random seeds. As shown in Table 9, our proposed method, EAR, achieves consistent improvements across both metrics compared to standard BC and Recovery on Pretrained Agent. However, we observe that for certain styles (e.g., Style B and C), simple behavior cloning can already achieve excellent performance. This is largely due to the fact that partners in this benchmark exhibit strong adaptability—a favorable condition that is not commonly observed and has been refuted by results in Style A and the Coordination Ring layout. Overall, our method demonstrates robust and competitive performance across diverse settings, highlighting its effectiveness under more realistic and challenging scenarios.

Furthermore, when cooperating with proxy agents that are distinct from the "like" trajectories in the benchmark, we investigate the generalization capability of collaborative agents. Table 10 reveals that while standard BC leads to overfitting on offline trajectories—resulting in degraded performance on out-of-distribution scenarios—our method maintains its effectiveness. This finding underscores the importance of the recovery in enabling generalization.

E Limitations

Our current work has certain limitations. Specifically, we have not fully leveraged the unlabeled trajectories left by human players, which are often more abundant (potentially in the hundreds) than the explicitly "liked" ones. While these trajectories may not directly reflect human satisfaction, they can still serve as suboptimal data. By assessing the distance between these unlabeled trajectories and the preferred ones, the collaborative agent may further improve its ability to learn and align with human preferences. Exploring how to incorporate such implicit human behavior signals represents a promising direction for future work.

F Compute resources

All experiments were conducted on a single NVIDIA GeForce RTX 2080 Ti GPU. Table 11 reports the per-run execution time with the number of rollouts set to 200. We focus on the results for the Many Orders layout under Style A.

Table 11: Compute resources

Methods	Execution time (s)
1-epoch Behavior Cloning (BC)	27.1
One round of environment rollouts	5.7
PPO (15 epochs) after one rollout	10.5