FROM TASK-ORIENTED AGENTS TO PERSONALIZED TEAMMATES: UNDERSTANDING CONTINUAL ADAPTATION IN COOPERATIVE GAMES

Anonymous authors
Paper under double-blind review

ABSTRACT

In cooperative games, task performance alone does not determine whether an AI teammate is a good fit for a player. Motivated by the data available to game developers, we study personalization from abundant gameplay logs and sparse, noisy post-match endorsements, without requiring the target player to participate in training. Gameplay logs describe how the player acts, whereas endorsements provide imperfect evidence of how they want a teammate to cooperate. This creates a continual adaptation challenge: preference learning and subsequent task optimization pursue different objectives and can interfere with one another. Through controlled experiments in *Overcooked*, we isolate the effects of preference initialization, training partners, and reward design. Our analysis shows that, in multi-agent on-policy learning, task optimization can lead to different behavioral adaptations under different training-partner distributions. The same task objective can improve a learned division of labor under one interaction condition, yet replace it under another. Guided by this analysis, we initialize the AI from endorsements, learn an interactive player proxy from gameplay logs, and continue task-reward RL with that proxy. This simple procedure improves both task performance and preference satisfaction in our benchmark without requiring preference reward.

1 INTRODUCTION

In multiplayer games such as *Overcooked* and *Honor of Kings*, a player's experience depends on both the outcome and how teammates work together. For developers building AI teammates, effective play is therefore only part of the goal; the AI should also cooperate in ways that individual players prefer. We consider a platform-side setting where historical player data are available, but the target player is unavailable for interactive training. The AI instead practices with simulated teammates before cooperating with that player at deployment. Cooperating with partners not encountered during training is a central problem in ad hoc teamwork (Stone et al., 2010) and zero-shot coordination (ZSC) (Hu et al., 2020; Mirsky et al., 2022). Population-based methods train agents with a diverse pool of simulated teammates to improve generalization to unfamiliar partners (Strouse et al., 2021; Yu et al., 2023). However, without accounting for the target player's behavior, even a diverse pool may leave the AI unprepared for out-of-distribution (OOD) interactions at deployment. Moreover, maximizing environmental reward can produce a division of labor that the player does not prefer. For example, a player may want an AI that supports their role rather than takes over, even if taking over would increase the task score (Figure 1). We study how to personalize a task-trained AI teammate while retaining its ability to complete the task.

Game developers can use existing gameplay records and player feedback to personalize AI teammates. Platforms can record the actions of players and their teammates, while games such as *League of Legends* and *Brawl Stars* provide post-match commendation features. A like marks the corresponding teammate trajectory as endorsed. This leaves developers with extensive gameplay records but relatively few explicit endorsements from each player. The distinction between player and teammate behavior matters here: the player's own actions describe how they play, while their endorsements concern how others cooperate with them. A player may expect an AI teammate to take a different role from their own, so copying the player's actions does not necessarily satisfy that expectation. In practice, the endorsement labels can also be misleading: a player may use a like sarcastically to

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



Figure 1: Platform-side personalization where player behavior and preferred teammate behavior differ, using gameplay logs and sparse post-match feedback.

criticize poor performance rather than express approval.[1] EAR (Ma et al., 2026) provides an initial study of personalization from sparse endorsements without online user interaction. We build on this foundation by considering the broader information available to a game platform: behavioral histories that reveal how the player acts, and limited, potentially unreliable feedback about the teammate behavior they prefer.

Existing methods learn user preferences through reward learning or direct imitation. Reward-learning methods use demonstrations or preference feedback to learn a reward function for policy optimization (Arora & Doshi, 2021; Wirth et al., 2017; Liu et al., 2025; Cheng et al., 2024). With only a few noisy endorsements, however, the available supervision may be insufficient for reliable reward estimation. Direct imitation, implemented through SFT (Foster et al., 2024; Chu et al., 2025), instead trains the agent to reproduce actions in endorsed trajectories. However, fitting a small set of these trajectories can reduce the task performance of an already capable agent. EAR (Ma et al., 2026) addresses this problem by alternating imitation with RL against a pool of simulated teammates. The imitation stage learns preferred actions, while the RL stage uses environmental rewards to restore task performance. Yet a higher task score can come from taking over work intended for the player, rather than performing the preferred role more effectively. We observe this behavior during EAR's recovery, even when action accuracy on endorsed trajectories remains high. The challenge is therefore to improve execution without undoing the cooperation learned from user feedback.

This challenge also arises in continual learning, where learning a new objective can change previously acquired behavior. Recent studies of language-model post-training examine this question by analyzing on-policy data and changes in the policy distribution (Chen et al., 2026; Shenfeld et al., 2026b). They consider both how training data are generated and how much adaptation changes the initial policy. These perspectives also inform our analysis of game-agent personalization, where preference fitting is followed by task-oriented learning. In cooperative games, the data-generation process involves both the learner and its teammate. During on-policy training, the learner samples actions from its current policy. Its teammate also affects the states it encounters and the task feedback it receives. Following these studies, we analyze how task-oriented RL changes behavior learned from endorsements. We consider both the learning procedure and the training partners.

To guide the design of personalized AI teammates, we analyze how preference initialization, training partners, and reward design affect adaptation in *Overcooked*. Controlled comparisons separate the effects of learning from endorsements from those of subsequent interaction. The results show that task

[1] See Supercell's description of Kudos and a player's account of sarcastic use.

rewards can improve both task performance and preference satisfaction after preference fine-tuning, without an additional preference reward. This benefit depends on the training partner; on-policy updates alone do not ensure it. Guided by these findings, PROXY-RL combines preference initialization from endorsements with task-reward RL against a player proxy learned from gameplay logs. This makes the procedure applicable without further interaction with the target player. Together, the analysis and implementation show how task feedback can complement limited preference supervision in training personalized teammates.

2 RELATED WORK

2.1 PARTNER GENERALIZATION IN HUMAN–AI COORDINATION

ZSC and ad hoc teamwork study how agents cooperate with partners they have not trained with. Population-based methods train agents against diverse simulated teammates, using self-play checkpoints, diversity objectives, and hidden reward functions to broaden the range of partner behaviors encountered during training (Strouse et al., 2021; Zhao et al., 2023; Yu et al., 2023). Further work explores adaptive partner generation and adversarial diversity (Ruhdorfer et al., 2025; Lauffer et al.), as well as partner-conditioned adaptation and policy transfer at deployment (Zhang et al., 2025; Ding et al., 2025; Li et al., 2025; Long et al., 2025; Nigam et al., 2025). Human behavior models fitted to gameplay data provide another source of training partners (Carroll et al., 2019b). A diverse simulated population may still omit behaviors exhibited by a target player, particularly when training does not use data from that player. Successful task coordination also differs from personalization: an agent may achieve high returns with a player while taking on responsibilities that the player would prefer to retain. These are separate concerns about partner coverage and the objective of cooperation. Our analysis considers training partners as a factor in personalization, examining how they affect the behavior retained during subsequent task-oriented learning.

2.2 PREFERENCE LEARNING AND SEQUENTIAL PERSONALIZATION

Personalizing a task-trained agent requires learning user-preferred behavior while retaining useful task skills. Reward-learning methods learn reward functions from demonstrations (Fu et al., 2017; Arora & Doshi, 2021) or preference feedback (Wirth et al., 2017; Lee et al., 2021; Abdelkareem et al., 2022; Gong et al., 2025). These rewards guide policy optimization in alignment methods such as RLHF (Achiam et al., 2023; Wang et al., 2024). Other methods optimize policies directly from preference data without training a separate reward model (Rafailov et al., 2023; Hejna et al., 2023; Kang & Oh). Beyond the choice of objective, policy adaptation also depends on how offline data and online interaction are used during training. Relevant RL methods include learning from fixed datasets (Kumar et al., 2020; Prudencio et al., 2023), offline-to-online fine-tuning (Nakamoto et al., 2023; Wang et al., 2023; Lee et al., 2022; Guo et al., 2023), online-to-offline personalization (Liu et al., 2025), and hybrid training with offline and online experience (Song et al., 2022; Ball et al., 2023). These approaches offer different ways to use existing data and further interaction, without requiring additional user feedback. For cooperative personalization, EAR (Ma et al., 2026) alternates imitation of endorsed trajectories with task-reward RL using a pool of simulated teammates. In our work, we study how game platforms can personalize AI teammates using gameplay logs and sparse, noisy endorsements. The logs describe the player's behavior, whereas the endorsements concern the teammate behavior they prefer. We study the interaction between task-reward optimization and preference learning to guide a simple personalization procedure.

2.3 CONTINUAL LEARNING AND POST-TRAINING DYNAMICS

Continual learning studies how models acquire new skills while retaining existing capabilities. Earlier methods reduce forgetting by constraining parameter updates, preserving earlier predictions, or reusing past experience (Kirkpatrick et al., 2017; Li & Hoiem, 2016; Lopez-Paz & Ranzato, 2017; Rolnick et al., 2019). Related analyses examine skill loss during RL fine-tuning (Wolczyk et al., 2024) and reduced ability to learn new behaviors after continued training (Lyle et al., 2023; Dohare et al., 2024). Language-model post-training raises similar questions about learning new tasks and preserving previously acquired skills and aligned behavior (Scialom et al., 2022; Qi et al., 2024). Recent work studies how SFT and RL differ in generalization (Chu et al., 2025) and investigates the causes of their

different forgetting patterns (Chen et al., 2026; Shenfeld et al., 2026b). Using controlled experiments and simplified models, these analyses examine on-policy data, changes in policy distributions, and optimization choices. They assess both performance on the new task and retention of previously learned capabilities. The role of data generation is also explored through on-policy self-distillation, which learns from demonstrations using the model's own sampled trajectories without requiring an explicit reward (Shenfeld et al., 2026a). These studies provide an analytical basis for understanding how successive training stages affect learned behavior. We adopt this approach to study the interaction between preference learning and task-reward optimization in cooperative game personalization.

3 Problem Setting

We study how a game platform can personalize a task-trained AI policy π_0 while maintaining task performance. Training uses historical player data and interactions with simulated teammates in the game environment, without online access to the target player.

Cooperative interaction. We model interaction between the AI and its partner as a two-player cooperative Markov game (Littman, 1994), with state space S, action spaces A^A and A^H, transition kernel T, initial-state distribution ρ, and shared task reward r_{env}. Let π and μ denote the AI and partner policies. At state s_t, they independently sample $a_t^A \sim \pi(\cdot \mid s_t)$ and $a_t^H \sim \mu(\cdot \mid s_t)$, followed by $s_{t+1} \sim T(\cdot \mid s_t, a_t^A, a_t^H)$. For a fixed partner μ, averaging over its actions gives the transition kernel

$$T_\mu(s' \mid s, a^A) = \sum_{a^H \in A^H} \mu(a^H \mid s)\, T(s' \mid s, a^A, a^H). \tag{1}$$

We write $p(\tau \mid \pi, \mu)$ for the resulting trajectory distribution.

Task performance and user preference. Let π_u^H denote the target player's policy. For a trajectory τ, $R_{\text{env}}(\tau)$ measures task performance and $R^u_{\text{pref}}(\tau)$ scores the AI's behavior according to the user's preferences. We define their expected returns as

$$J_{\text{env}}(\pi, \mu) = \mathbb{E}_{\tau \sim p(\cdot \mid \pi, \mu)}[R_{\text{env}}(\tau)], \qquad J^u_{\text{pref}}(\pi) = \mathbb{E}_{\tau \sim p(\cdot \mid \pi, \pi_u^H)}[R^u_{\text{pref}}(\tau)].$$

Imitating π_u^H does not necessarily satisfy the player's expectations of the AI. We evaluate both returns with the target player, since higher task return need not imply greater preference satisfaction. The true preference score is not available to the training algorithm.

Offline user data. The platform records the target player's interactions with teammates in $\mathcal{D}_{\log}$, including both players' actions. Post-match likes mark a small subset of these trajectories as endorsed; we denote this subset by $\mathcal{D}_{\text{like}}$. The endorsements concern the teammate's behavior, rather than the target player's own actions. To account for incorrect endorsements, we write

$$\mathcal{D}_{\text{like}} = \mathcal{D}_{\text{clean}} \cup \mathcal{D}_{\text{noise}}, \tag{2}$$

where the two subsets contain correctly and incorrectly endorsed trajectories, respectively. The training algorithm receives the endorsement labels, but not the clean/noisy partition.

Available training interactions. The AI can collect additional experience with existing or newly trained teammate policies in the game environment. Let q denote the distribution over these partner policies. The expected task return during training is

$$J^q_{\text{env}}(\pi) = \mathbb{E}_{\mu \sim q}[J_{\text{env}}(\pi, \mu)]. \tag{3}$$

At evaluation, the AI is paired with the target player π_u^H, and both task and preference returns are reported.

Environment and dataset construction. We conduct our experiments in the two-player cooperative game `Overcooked` (Ma et al., 2026; Yu et al., 2023). The benchmark includes `Many Orders`, `Coordination Ring`, and `Distant Tomato`, with three synthetic preference styles in each layout. Each style is specified by a weighted reward over gameplay events and task outcomes (Appendix B). A neutral policy represents the target player, and a reference teammate is trained

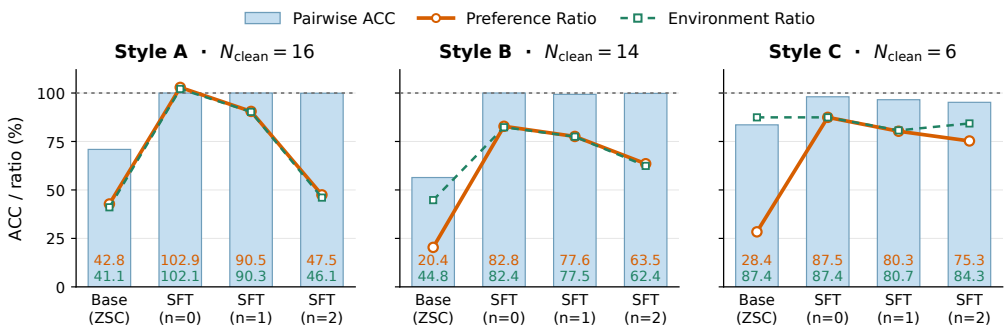

Figure 2: SFT in `Many Orders` with N_{clean} clean and n incorrectly endorsed trajectories. Bars show pairwise discrimination accuracy; lines show preference and task returns relative to their clean-reference means (100%). Base denotes the policy before preference SFT.

to optimize the corresponding preference reward with that player. We collect the target player's interactions with different teammates in $\mathcal{D}_{\log}$. We select clean endorsed trajectories from $\mathcal{D}_{\log}$ using a preference-return threshold. Noisy feedback additionally includes false-positive endorsements assigned to other trajectories from the same logs that do not match the target preference. For the `Many Orders` analysis, we consider $n = |\mathcal{D}_{\text{noise}}| \in \{1, 2\}$. Even these small counts can represent a substantial fraction of a sparse endorsement set. For example, two incorrect endorsements alongside six clean trajectories yield a 25% noise rate. The training algorithm does not receive the clean/noisy labels. Before personalization, we train π_0 with FCP (Strouse et al., 2021), a population-based ZSC method, using only the task reward. Environment details are provided in Appendix A; preference rewards, policy selection, trajectory filtering, and dataset sizes are specified in Appendix B.

Evaluation. For each layout–style pair, all policies are evaluated with the same target player. Preference return measures how well their behavior matches the specified style. We also report task return to measure progress toward the game's objectives. Both metrics are computed from the same evaluation trajectories. The main results are reported as means and standard deviations across three training seeds. Rollout settings and checkpoint selection are detailed in Appendix C.7.

4 Understanding Sequential Personalization

Inspired by recent analyses of LLM post-training (Chen et al., 2026), we investigate the factors that affect the personalization of task-trained game agents. We implement the reinforcement learning (RL) stages with multi-agent proximal policy optimization (MAPPO) (Yu et al., 2022), as in HSP and EAR (Yu et al., 2023; Ma et al., 2026). We first ask whether SFT on the available endorsements is sufficient for personalization. We compare clean and noisy feedback to assess how its effectiveness depends on feedback quality. We then examine whether further RL improves user-preferred behavior and how its effects vary with the policy initialization and training partners. This analysis also tests whether an explicit preference reward provides benefits beyond task reward alone. Finally, we use these findings to guide a simple training procedure based on player logs and endorsements, without requiring the target player to participate in training.

4.1 Preference Acquisition under Noisy Feedback

As a direct personalization baseline, we apply SFT to the task-trained policy using teammate actions from endorsed trajectories. Figure 2 compares the base policy with SFT on $\mathcal{D}_{\text{clean}}$ and on $\mathcal{D}_{\text{like}}$ with $n = |\mathcal{D}_{\text{noise}}| \in \{1, 2\}$. To assess whether unreliable feedback affects offline discrimination and actual execution differently, we evaluate both trajectory scoring and cooperative play. For scoring, we use the policy log-likelihood $LL_\pi(\tau) = \sum_t \log \pi(a_t^A \mid s_t)$. Discrimination accuracy is the fraction of evaluated clean–background trajectory pairs in which the clean trajectory receives the higher score.

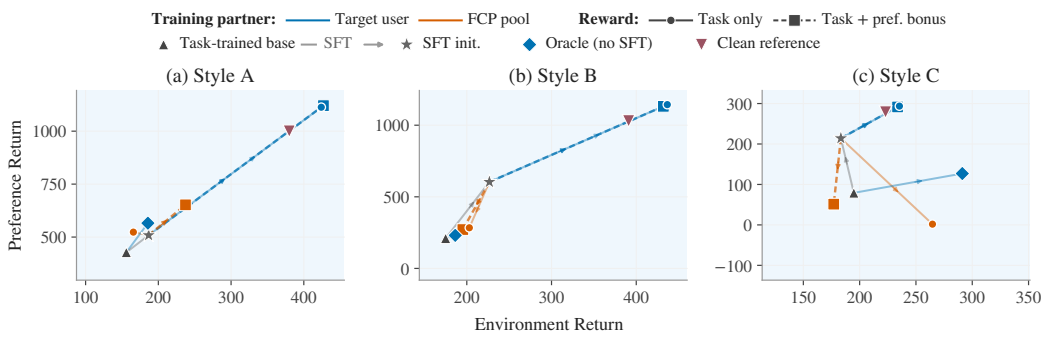

Figure 3: Effects of initialization, training partners, and rewards on personalization in `Many Orders`. Solid arrows connect training-stage endpoints; dashed arrows compare independently trained policies with and without the preference bonus.

Data efficiency does not imply noise robustness. With clean endorsements, SFT substantially improves preference performance over the task-trained baseline using only a few trajectories. This gain is not explained by increased task progress alone: in Style C, preference performance improves while task return remains unchanged. A small number of incorrectly endorsed trajectories can nevertheless undermine these gains. Preference performance declines across all three styles under noisy feedback, with the sharpest degradation in Style A. These results identify feedback reliability as a practical constraint on SFT-based personalization: learning effectively from few clean examples does not ensure stable performance when some endorsements are incorrect.

Preference recognition without reliable execution. Under noisy feedback, pairwise likelihood accuracy remains high while preference performance during play declines. The SFT policy still assigns higher scores to clean preferred trajectories than to most background trajectories. This provides evidence that preference-related information remains encoded in its trajectory scores. Appendix G explains how these outcomes can coexist under localized noise. In a tabular SFT model, conflicting labels reduce the probability of choosing the required action at each affected state. Executing the preferred behavior requires a sequence of coordinated decisions, so these errors can accumulate. By contrast, the same noisy trajectories may carry little weight in pairwise accuracy when compared against a large background archive. If clean trajectories remain distinguishable from the rest of that archive, overall ranking accuracy can stay high despite failures to separate clean examples from mislabeled ones. Despite noisy feedback, the SFT policy retains a preference-related likelihood ordering on the recorded trajectories. This distinction does not consistently translate into user-preferred behavior during interaction. We next test whether further task-oriented interaction can improve execution from this preference-informed starting point.

4.2 On-Policy Adaptation: Initialization, Partners, and Rewards

We test whether task-reward RL can improve cooperation after preference SFT. Because the AI and its partner jointly generate training experience, we examine both the starting policy and the training partner. Figure 3 first compares task-only RL from the base and noisy-SFT policies with the same target-player policy. We then hold the noisy-SFT initialization fixed and compare training with this player and with an existing FCP partner pool. For these diagnostics, we use the known target-player policy directly, setting aside the problem of learning a player proxy from gameplay logs. Under each partner condition, we also compare task-only RL with a terminal preference bonus derived from a frozen SFT policy's trajectory log-likelihood. A trapezoidal transform assigns the same maximum bonus within the observed endorsement-score interval and decreases linearly to zero on either side. The bonus is derived from the SFT model rather than the benchmark preference score. Its construction, parameters, and theoretical motivation are provided in Appendix C.1 and I. Diagnostic configurations are provided in Appendix C.9.

Initialization and partners shape on-policy adaptation. With the target user and task reward held fixed, preference-SFT initialization yields higher preference returns than initialization from the

task-trained base across all three styles (Figure 3). The endorsements provide guidance about the AI's expected role that interaction with the player alone does not supply. Starting from the same SFT checkpoint, task-only RL with the target user improves both returns across the three styles. Training with the FCP pool yields lower preference returns, with losses relative to the SFT starting point in Styles B and C. Both conditions use the same on-policy update rule, yet differ in preference retention. This reveals a partner-dependent aspect of on-policy learning in cooperative games. Sampling actions from the current AI policy does not fully characterize the experience used for adaptation: the partner also shapes the interaction in which those actions receive task feedback. Our comparisons show that this interaction context matters for retaining behavior introduced through SFT. The target-user results further show that preference supervision can influence later learning through the initialization, even when RL uses only task reward. SFT biases the policy toward the user's preferred collaboration style. With a suitable training partner, task feedback can then help the AI perform that style more effectively. The resulting personalization does not require task reward to identify the user's preference from scratch: it builds on the preference already introduced into the policy.

Limited gains from the preference surrogate. Adding the preference surrogate leaves both returns largely unchanged under target-user training. Under FCP training, Style A improves on both metrics, while the other styles show performance declines or preference gains at the cost of task return. The improvement in Style A shows that the surrogate can provide useful guidance, but this benefit does not extend consistently across the tested styles. Overall, adding the preference bonus to FCP training remains insufficient to achieve the preference performance obtained by training with the target user.

Action accuracy and preference retention. Recent LLM post-training studies examine how policy changes affect previously learned capabilities (Chen et al., 2026; Shenfeld et al., 2026b). We test whether maintaining agreement with endorsed actions during subsequent task-oriented RL also preserves user-preferred cooperation. EAR (Ma et al., 2026) provides a relevant test: it monitors preference recovery through action accuracy and alternates SFT with task-reward RL.

We measure action accuracy as the fraction of recorded AI actions predicted correctly on the clean endorsement data. Table 1 compares EAR with Proxy-RL. In Style B, EAR and Proxy-RL differ by only 0.48 percentage points in mean action accuracy, but their preference ratios are 86.12% for EAR and 108.95% for Proxy-RL. Similar agreement with demonstrated actions therefore coexists with different preference performance. The checkpoint curves in Appendix H further compare changes in action accuracy and rollout performance during adaptation. Maintaining action accuracy alone is thus insufficient to assess whether task-oriented learning preserves the intended cooperation.

Table 1: Action fit and preference performance. ACC gap is Proxy-RL minus EAR on clean endorsements, in percentage points. Preference ratios list EAR / Proxy-RL as percentages of the mean preference return of clean reference trajectories.

Style	ACC gap	Pref. ratio (%)
A	13.29	68.88 / 108.37
B	0.48	86.12 / 108.95
C	12.03	46.23 / 105.97

The event statistics make this distinction concrete. In `Many Orders` Styles A and B, the clean reference trajectories leave soup delivery to the player. Proxy-RL preserves this pattern, whereas EAR performs about two deliveries per episode (Table 14). Proxy-RL nevertheless achieves higher task returns. Thus, improved task progress does not require the AI to take over delivery duties. These results provide behavioral evidence that effective cooperation can preserve the reference division of responsibility.

4.3 Personalization with a Learned User Proxy

The analysis above shows that subsequent task learning depends on both the AI's preference initialization and its training partner. We use the data available to a game platform to implement this combination. Gameplay logs $\mathcal{D}_{\text{log}}$ provide examples of the target player's actions for training a proxy, while endorsed trajectories $\mathcal{D}_{\text{like}}$ provide the AI's preference supervision. The AI then continues learning with the proxy using the environmental task reward. We call this procedure Proxy-RL (Figure 4).

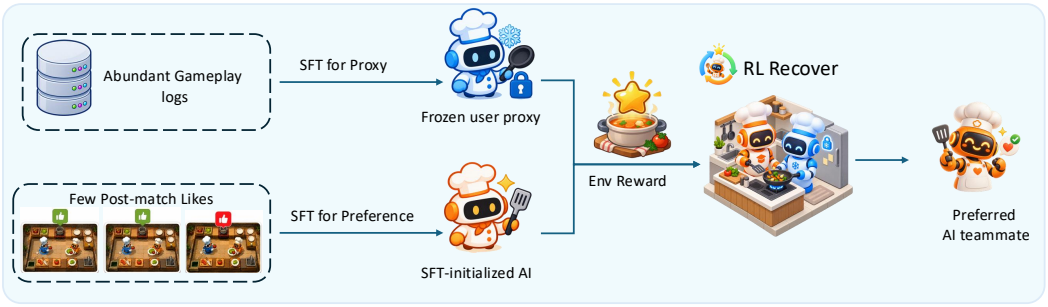

Figure 4: Proxy-RL combines preference SFT with task-reward RL using a player proxy learned from gameplay logs.

Learning the training partner. We fit a proxy policy $\hat{\mu}_\psi$ by SFT on the target player's state–action pairs from the abundant gameplay logs:

$$\mathcal{L}_{\text{proxy}}(\psi) = \mathbb{E}_{(s,a^H)\sim\mathcal{D}_{\log}}\left[-\hat{\mu}_\psi(a^H \mid s) + \log \sum_{b\in A^H} \exp\big(\hat{\mu}_\psi(b \mid s)\big)\right]. \tag{4}$$

After fitting, we freeze the proxy's parameters. During RL recovery, the proxy samples actions from its learned policy and cooperates with the AI. The current AI policy and this fixed partner jointly generate the on-policy rollouts used to update the AI. Training details are provided in Appendix C.5.

Task-oriented adaptation. We apply SFT to the task-trained policy π_0 using endorsed teammate actions in $\mathcal{D}_{\text{like}}$ to obtain π_{SFT}. We initialize the AI with this policy and update only its parameters θ during interaction with the proxy. The recovery objective uses the environmental reward, with no additional preference bonus:

$$\max_\theta J_{\text{env}}(\pi_\theta, \hat{\mu}_\psi), \qquad \pi_{\theta_0} = \pi_{\text{SFT}}. \tag{5}$$

SFT and RL serve complementary roles in this continual learning process. SFT introduces the user's preferred collaboration style, and the learned player proxy provides a partner for further practice. Task feedback guides improvements in execution, while the SFT initialization carries preference information into subsequent learning. The procedure thus builds on earlier preference learning without requiring preference labels for the new rollouts. The next section evaluates task performance and preference satisfaction across the benchmark.

5 Experiments

We evaluate Proxy-RL across the benchmark to assess its task performance and preference satisfaction relative to existing methods.

Baselines and evaluation. FCP (Strouse et al., 2021) provides the task-trained baseline, using diverse partners to improve zero-shot coordination without preference supervision. We also evaluate BC/SFT under clean and noisy feedback, and CPL (Hejna et al., 2023) as an offline preference-learning baseline, with implementation details in Appendix C.6. EAR (Ma et al., 2026) alternates imitation of clean endorsements with task-reward RL against a partner pool. Oracle starts from the task-trained policy and performs task-reward RL directly with the target-user policy, without preference SFT. Alongside the default Proxy-RL, we report Proxy-RL+Pref., which adds the likelihood-based preference bonus to the same training procedure. We use the Environment Return and Preference Score defined in Section 3 and follow the evaluation protocol specified there. Table 2 summarizes the results on `Many Orders`.

Overall performance. Table 2 shows that the Proxy-RL variants achieve the highest mean Preference Scores among the evaluated methods across all three `Many Orders` styles. The default

Table 2: Performance on `Many Orders`, reported as mean ± standard deviation over three training seeds and rounded to three significant figures. $n = |\mathcal{D}_{\text{noise}}|$ counts incorrectly endorsed trajectories; All PROXY-RL configurations use $n = 2$. Bold indicates the highest method mean in each column. Additional layouts are reported in Appendix D.

Method	**Style A**		**Style B**		**Style C**	
	Pref. Score	Env. Return	Pref. Score	Env. Return	Pref. Score	Env. Return
Offline trajectories	1000	380	1030	391	279	223
FCP (ZSC)	428 ± 40.7	156 ± 20.5	210 ± 164	175 ± 5.89	79.2 ± 72.4	195 ± 64.1
Oracle	566 ± 135	186 ± 0.943	231 ± 186	186 ± 3.09	127 ± 96.7	$\mathbf{291} \pm 42.8$
EAR	689 ± 231	241 ± 96.4	887 ± 245	335 ± 93.2	129 ± 94.7	220 ± 40.3
SFT ($n = 0$)	1030 ± 2.30	388 ± 0.816	854 ± 353	322 ± 137	244 ± 29.3	195 ± 23.5
SFT ($n = 1$)	906 ± 247	343 ± 96.0	800 ± 245	303 ± 95.0	224 ± 58.0	180 ± 46.0
SFT ($n = 2$)	475 ± 461	175 ± 179	655 ± 341	244 ± 134	210 ± 69.0	188 ± 36.0
CPL	21.3 ± 21.1	1.00 ± 1.41	18.5 ± 26.2	6.67 ± 9.43	28.6 ± 40.4	23.0 ± 32.5
PROXY-RL	1080 ± 36.9	412 ± 15.6	1120 ± 22.0	428 ± 8.60	$\mathbf{296} \pm 1.01$	237 ± 0.471
with entropy-regularized SFT	1100 ± 14.2	417 ± 4.50	$\mathbf{1130} \pm 13.3$	$\mathbf{432} \pm 4.92$	288 ± 4.45	232 ± 4.03
with checkpoint pool ($K = 3$)	$\mathbf{1120} \pm 10.4$	$\mathbf{425} \pm 4.90$	1120 ± 17.8	426 ± 6.68	295 ± 2.04	236 ± 1.63
PROXY-RL+PREF.	1040 ± 102	395 ± 39.2	1120 ± 12.1	428 ± 4.92	282 ± 6.52	228 ± 3.77

SFT-fitted proxy performs close to the best variant in each style, while the reward-augmented version also performs well. SFT with clean endorsements remains competitive, particularly in Style A, but degrades under noisy feedback. Both proxy-based variants achieve strong preference performance in the noisy setting. Together with the controlled comparisons in Section 4.2, these results highlight the roles of preference initialization and user-related training interactions. Task-reward RL provides the main gains, while clear benefits from the additional preference reward appear in only a few evaluated cases. For example, under stronger task–preference conflict in `Distant Tomato`, the bonus improves preference scores at the cost of task return (Table 13). Moreover, in `Many Orders` Style C, Oracle achieves the highest task return but much lower preference performance than Proxy-RL. This contrast shows that the preference gains are not simply a consequence of increased task progress.

The benefit also extends beyond `Many Orders`. On `Coordination Ring`, Proxy-RL achieves higher mean task and preference returns than SFT under the same noisy-feedback condition in all three styles (Appendix D). On `Distant Tomato`, its preference scores approach the clean-reference values in Styles A and B. Style C presents a different outcome: the reference trajectories achieve only 10 task points per episode, and neither proxy-based variant matches their preference score. The additional bonus improves preference return but reduces task return (Table 13).

Ablation on proxy fitting. Given the abundant gameplay logs, we test whether a single SFT-trained proxy is sufficient for subsequent personalization. We compare it with two alternatives: SFT with entropy regularization and a pool of three checkpoints from proxy training, inspired by FCP (Strouse et al., 2021). The former encourages more stochastic actions within one proxy, while the latter exposes the AI to different partner policies during RL recovery. Table 2 shows that the single-proxy baseline achieves task and preference performance close to both alternatives. The limited gains from these additional sources of diversity support using a single SFT-fitted partner with the available player logs. Construction details and additional results are provided in Appendix C.8.

6 CONCLUSION

Motivated by the data available to game platforms, we studied personalization from gameplay logs and sparse, noisy endorsements. Through controlled experiments in *Overcooked*, we analyzed the roles of preference initialization, training partners, and rewards in adapting task-trained agents. The results show that task-reward RL can improve user-preferred behavior after SFT, but that the training partner influences whether this behavior is retained or replaced. These findings led to PROXY-RL, which combines preference SFT with task-reward RL using a player proxy learned from logs. Experiments show that this simple procedure can improve both task performance and preference satisfaction under noisy feedback, without an additional preference reward or direct interaction with the target player. Our findings emphasize that continual personalization requires attention to both the behavior acquired from user feedback and the training interactions that support its further development.

AI USE STATEMENT

We used AI tools to assist with manuscript drafting, language editing, and searches for related literature. The authors take full responsibility for the final manuscript, including the accuracy of its references, scientific claims, and interpretation of the results.

ETHICS STATEMENT

This study uses simulated players and synthetically constructed preference feedback in a cooperative game environment. No human participants were recruited, and no personal player data were collected for this study. The experiments concern controlled simulations and do not establish effectiveness or safety in real-player deployments. Applications involving real gameplay logs would require appropriate safeguards for player privacy and consent.

REPRODUCIBILITY STATEMENT

The main text describes the problem formulation, learning procedures, and evaluation metrics. The appendix provides further details on benchmark construction, data generation and feedback contamination, training configurations, and evaluation protocols. Additional results and implementation details are included to support reproduction and interpretation of the reported findings.

REFERENCES

Youssef Abdelkareem, Shady Shehata, and Fakhri Karray. Advances in preference-based reinforcement learning: A review. In *2022 IEEE international conference on systems, man, and cybernetics (SMC)*, pp. 2527–2532. IEEE, 2022.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023.

Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. *On the utility of learning about humans for human-AI coordination*. Curran Associates Inc., Red Hook, NY, USA, 2019a.

Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human–ai coordination. In *Advances in Neural Information Processing Systems*, volume 32, 2019b.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. Retaining by doing: The role of on-policy data in mitigating forgetting. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2510.18874.

Jie Cheng, Gang Xiong, Xingyuan Dai, Qinghai Miao, Yisheng Lv, and Fei-Yue Wang. RIME: Robust preference-based reinforcement learning with noisy preferences. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8229–8247, 2024. URL https://proceedings.mlr.press/v235/cheng24k.html.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10818–10838. PMLR, 2025. URL https://proceedings.mlr.press/v267/chu25c.html.

Hao Ding, Chengxing Jia, Zongzhang Zhang, Cong Guan, Feng Chen, Lei Yuan, and Yang Yu. Learning to coordinate with different teammates via team probing. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632: 768–774, 2024. doi: 10.1038/s41586-024-07711-7.

Dylan J. Foster, Adam Block, and Dipendra Misra. Is behavior cloning all you need? understanding horizon in imitation learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL `https://arxiv.org/abs/2407.15007`.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Ze Gong, Akshat Kumar, and Pradeep Varakantham. Offline safe reinforcement learning using trajectory classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 16880–16887, 2025.

Siyuan Guo, Lixin Zou, Hechang Chen, Bohao Qu, Haotian Chi, Philip S Yu, and Yi Chang. Sample efficient offline-to-online reinforcement learning. *IEEE Transactions on Knowledge and Data Engineering*, 36(3):1299–1310, 2023.

Joey Hejna, Rafael Rafailov, Harshit Sikchi, Chelsea Finn, Scott Niekum, W Bradley Knox, and Dorsa Sadigh. Contrastive preference learning: learning from human feedback without rl. *arXiv preprint arXiv:2310.13639*, 2023.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play " for zero-shot coordination. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Hyungkyu Kang and Min-hwan Oh. Offline preference-based value optimization. In *The Fourteenth International Conference on Learning Representations*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Niklas Lauffer, Ameesh Shah, Micah Carroll, Sanjit A Seshia, Stuart Russell, and Michael D Dennis. Robust and diverse multi-agent learning via rational policy gradient. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pp. 1702–1712. PMLR, 2022.

Benjamin Li, Shuyang Shi, Lucia Romero, Huao Li, Yaqi Xie, Woojun Kim, Stefanos Nikolaidis, Michael Lewis, Katia Sycara, and Simon Stepputtis. Modeling latent partner strategies for adaptive zero-shot human-agent collaboration. *arXiv preprint arXiv:2507.05244*, 2025.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *Computer Vision – ECCV 2016*, volume 9908 of *Lecture Notes in Computer Science*, pp. 614–629. Springer, 2016. doi: 10.1007/978-3-319-46493-0_37.

Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 157–163. Morgan Kaufmann, 1994. doi: 10.1016/B978-1-55860-335-6.50027-1.

Xu Liu, Haobo Fu, Stefano V Albrecht, QIANG FU, and Shuai Li. Online-to-offline RL for agent alignment. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ruv3HdK6he.

Weifan Long, Wen Wen, Peng Zhai, and Lihua Zhang. Role play: Learning adaptive role-specific strategies in multi-agent interactions. *Knowledge-Based Systems*, 324:113811, 2025.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper/2017/hash/f87522788a2be2d171666752f97ddebb-Abstract.html.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23190–23211. PMLR, 2023. URL https://proceedings.mlr.press/v202/lyle23b.html.

Haitong Ma, Haoran Yu, Haobo Fu, and Shuai Li. Learning preferences without interaction for cooperative AI: A hybrid offline-online approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=QVheAhJefR.

Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano V. Albrecht. A survey of ad hoc teamwork research. In *Multi-Agent Systems: 19th European Conference, EUMAS 2022, Düsseldorf, Germany, September 14–16, 2022, Proceedings*, pp. 275–293, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-20613-9. doi: 10.1007/978-3-031-20614-6_16. URL https://doi.org/10.1007/978-3-031-20614-6_16.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

Rupal Nigam, Niket Parikh, Hamid Osooli, Mikihisa Yuasa, Jacob Heglund, and Huy T Tran. Zero-shot coordination in ad hoc teams with generalized policy improvement and difference rewards. *arXiv preprint arXiv:2510.16187*, 2025.

Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/83b7da3ed13f06c13ce82235c8eedf35-Abstract-Conference.html.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Abstract.html.

Constantin Ruhdorfer, Matteo Bortoletto, Victor Oei, Anna Penzkofer, and Andreas Bulling. Unsupervised partner design enables robust ad-hoc teamwork. *arXiv preprint arXiv:2508.06336*, 2025.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6107–6122. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.410. URL https://aclanthology.org/2022.emnlp-main.410/.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026a. URL https://arxiv.org/abs/2601.19897.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less. In *International Conference on Learning Representations*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/618c95f4557c15b253fb0e6f548ea0c0-Abstract-Conference.html.

Yuda Song, Yifei Zhou, Ayush Sekhari, J Andrew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid rl: Using both offline and online data can make rl efficient. *arXiv preprint arXiv:2210.06718*, 2022.

Peter Stone, Gal A. Kaminka, Sarit Kraus, and Jeffrey S. Rosenschein. Ad hoc autonomous agent teams: collaboration without pre-coordination. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, pp. 1504–1509. AAAI Press, 2010.

DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. Collaborating with humans without human data. *Advances in Neural Information Processing Systems*, 34: 14502–14515, 2021.

Shenzhi Wang, Qisen Yang, Jiawei Gao, Matthieu Lin, Hao Chen, Liwei Wu, Ning Jia, Shiji Song, and Gao Huang. Train once, get a family: State-adaptive balances for offline-to-online reinforcement learning. *Advances in Neural Information Processing Systems*, 36:47081–47104, 2023.

Zhichao Wang, Bin Bi, Shiva Kumar Pentyala, Kiran Ramnath, Sougata Chaudhuri, Shubham Mehrotra, Xiang-Bo Mao, Sitaram Asur, et al. A comprehensive survey of llm alignment techniques: Rlhf, rlaif, ppo, dpo and more. *arXiv preprint arXiv:2407.16216*, 2024.

Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46, 2017.

Maciej Wolczyk, Bartłomiej Cupiał, Mateusz Ostaszewski, Michał Bortkiewicz, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Fine-tuning reinforcement learning models is secretly a forgetting mitigation problem. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53039–53078. PMLR, 2024. URL https://proceedings.mlr.press/v235/wolczyk24a.html.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24611–24624, 2022.

Chao Yu, Jiaxuan Gao, Weilin Liu, Botian Xu, Hao Tang, Jiaqi Yang, Yu Wang, and Yi Wu. Learning zero-shot cooperation with humans, assuming humans are biased. *arXiv preprint arXiv:2302.01605*, 2023.

Zuyuan Zhang, Hanhan Zhou, Mahdi Imani, Taeyoung Lee, and Tian Lan. Learning to collaborate with unknown agents in the absence of reward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14502–14511, 2025.

Rui Zhao, Jinming Song, Yufeng Yuan, Haifeng Hu, Yang Gao, Yi Wu, Zhongqian Sun, and Wei Yang. Maximum entropy population-based training for zero-shot human-ai coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6145–6153, 2023.

A OVERCOOKED: TASK OVERVIEW AND LAYOUT DETAILS

We utilize the `Overcooked` environment (Carroll et al., 2019a) as our testbed, which is a widely recognized benchmark for evaluating multi-agent cooperative RL. The domain simulates a dynamic grid-world kitchen where two agents must collaborate seamlessly under strict time constraints to prepare and serve soups, aiming to maximize a shared environmental return.

To successfully complete a delivery, the agents must coordinate on a sequence of strictly interdependent sub-tasks:

- Ingredient Assembly: Navigating the kitchen to retrieve raw ingredients and deposit them into the cooking pots.
- Cooking Phase: Allowing the assembled ingredients to cook for a predetermined duration.
- Plating and Serving: Transferring the finished soup onto a dish and delivering it to the designated dispatch counter.

At each timestep, agents select from a discrete action space consisting of four directional movements (up, down, left, right), a specialized interact action, and a no-op. Crucially, achieving optimal team performance in this confined spatial layout extends far beyond mastering the objective task sequence. It strictly demands proactive spatial coordination and the capacity to dynamically adapt to the latent behavioral styles and role-division preferences of the partner, making it useful for evaluating personalized coordination under different task and role preferences.

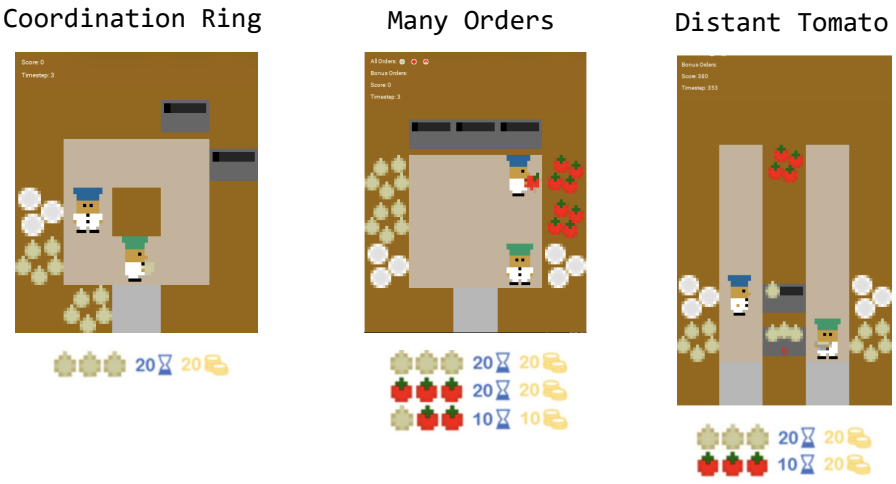

Figure 5: Three layouts in our experiments.

A.1 LAYOUTS

To ensure a rigorous and comparable evaluation, we adopt two established layouts from prior work (Ma et al., 2026), `Coordination Ring` and `Many Orders`, while introducing a novel layout, `Distant Tomato`, to further validate the generalized robustness of our framework across diverse environmental constraints.

As illustrated in the left panel of Figure 5, **Coordination Ring** features a continuous, loop-like kitchen topology. Because the objective is restricted to a single recipe (onion soup, which requires 20 cooking steps and yields 20 points per delivery), task-level decision-making is straightforward. Instead, the primary collaborative bottleneck is strictly spatial. All ingredients, plates, and serving stations are densely clustered in the bottom-left corner, while the cooking pots are isolated in the top-right. Consequently, agents must implicitly negotiate movement directions and yielding behaviors to prevent counterproductive collisions in the narrow corridors.

In contrast, `Many Orders` (Figure 5, center) utilizes an open, square-shaped kitchen that shifts the collaborative focus from spatial navigation to task-level prioritization. This environment introduces

significant recipe diversity, featuring pure onion soups, pure tomato soups, and a mixed recipe consisting of 1 onion and 2 tomatoes, each yielding different rewards. The core challenge here is strategic alignment: if the human partner demonstrates a stylistic preference for pursuing a specific recipe pipeline, the cooperative agent must rapidly identify and adapt to this choice to maintain a synchronized workflow.

Finally, the `Distant Tomato` layout (Figure 5, right) is introduced to test coordination in the absence of physical proximity. In this map, the two agents operate in strictly divided work areas, entirely eliminating the risk of movement collisions or spatial interference. The environment supports two distinct recipes: onion soup (requiring 20 cooking steps) and tomato soup (requiring only 10 cooking steps), with both yielding 20 points upon delivery. Because the physical barrier prevents agents from directly interfering with each other's movement, successful collaboration relies entirely on high-level cognitive alignment. The players must reach a unified consensus on which recipe to pursue to optimize the team's overall efficiency.

B Preference Design and Event Spaces

This section defines the preference rewards and describes how we select preference styles and simulated players for the benchmark.

Event-based preferences. Following prior work on stylized cooperation (Yu et al., 2023; Ma et al., 2026), we represent each preference configuration using weighted gameplay events. For a layout with d events, let $\phi(s, a^A, a^H) \in \mathbb{R}^d$ record their occurrence at a transition. A configuration (w, c_{env}) defines

$$r_{\text{pref}}(s, a^A, a^H) = \phi(s, a^A, a^H)^\top w + c_{\text{env}}\, r_{\text{env}}(s, a^A, a^H), \tag{6}$$

where w specifies the event weights and c_{env} weights the team's sparse environment return. Positive event weights favor the corresponding behavior, negative weights discourage it, and zero weights leave it unscored. The Preference Score of a trajectory is the sum of r_{pref} over the episode.

For each layout, we use 36 predefined preference configurations. Tables 3, 4, and 5 report the event dimensions and the weight values that occur in these configurations. The specific configurations selected as Styles A–C are reported in the following subsection.

Table 3: Definitions of sub-events and their candidate stylistic weights in the *Coordination Ring* layout. The final row specifies the coefficient of the additional team sparse environment return.

Sub-Event (e_i)	**Weight Values** (w_i)
Acquiring an onion from the dispenser	$\{-10, 0, 10\}$
Acquiring a clean dish from the dispenser	$\{0, 10\}$
Plating cooked soup directly from the pot	$\{-10, 0, 10\}$
Depositing an onion into the cooking pot	$\{-10, 0, 10\}$
Serving the plated soup to the delivery counter	$\{-10, 0\}$
Team sparse environment return (c_{env})	$\{0, 1\}$

B.1 Preference Vector Selection and Benchmark Construction

For each of the 36 preference configurations in a layout, we train a stylized policy alongside a neutral policy. The stylized policy optimizes the corresponding r_{pref}, while the neutral policy optimizes the environmental task reward. We select three target styles with large pairwise Euclidean distances between their weight vectors. If a selected style does not produce the intended combination of abundant gameplay logs and sparse endorsements, we replace it with another viable candidate using the same distance-based criterion.

Table 6 lists the selected configurations. Within each layout, the entries of $\mathbf{w}$ follow the event order in the corresponding table. The coefficient c_{env} is listed separately and weights the team sparse environmental reward. In `Many Orders`, the weight for `useless_placement` is included in $\mathbf{w}$.

Table 4: Definitions of sub-events and their candidate stylistic weights in the *Many Orders* layout. The final row specifies the coefficient of the additional team sparse environment return.

Sub-Event (e_i)	**Weight Values** (w_i)
Acquiring an onion from the dispenser	$\{-5, 0, 5\}$
Acquiring a clean dish from the dispenser	$\{-5, 0, 5\}$
Acquiring a tomato from the dispenser	$\{-5, 0, 5\}$
Retrieving a ready soup	$\{-5, 0, 5\}$
Executing a viable item placement	$\{-5, 0, 5\}$
Executing an optimal item placement	$\{-10, 0, 10\}$
Executing a catastrophic item placement	$\{-10, 0, 10\}$
Depositing an onion into the cooking pot	$\{-3, 0, 3\}$
Depositing a tomato into the cooking pot	$\{-3, 0, 3\}$
Serving the plated soup to the delivery counter	$\{-10, 0, 10\}$
Executing a useless item placement	$\{-1, 0, 1\}$
Team sparse environment return (c_{env})	$\{0, 1\}$

Table 5: Definitions of sub-events and their stylistic weight values in the *Distant Tomato* layout. The final row specifies the coefficient of the additional team sparse environment return.

Sub-Event (e_i)	**Weight Values** (w_i)
Acquiring an onion from the dispenser	$\{-5, 0, 5\}$
Acquiring a tomato from the dispenser	$\{0, 10, 20\}$
Acquiring a clean dish from the dispenser	$\{0, 10\}$
Retrieving a ready soup	$\{-5, 0, 5\}$
Executing a viable item placement	$\{-10, 0, 10\}$
Executing an optimal item placement	$\{-10, 0, 10\}$
Executing a catastrophic item placement	$\{0, 10\}$
Depositing an onion into the cooking pot	$\{-10, 0, 10\}$
Depositing a tomato into the cooking pot	$\{-10, 0, 10\}$
Serving the plated soup to the delivery counter	$\{-10, 0\}$
Team sparse environment return (c_{env})	$\{0, 1\}$

For each selected style, we evaluate the trained stylized–neutral policy pairs using the corresponding preference reward. We choose the pair with the highest evaluated preference return. Its neutral policy becomes the target player π_u^H, and its stylized policy serves as the reference teammate. The selected vector defines the preference criterion; the reference teammate provides example behavior for benchmark construction. The target player is then used for gameplay-log collection and subsequent policy evaluation.

For each target player, we collect two trajectories with each of the 36 trained stylized policies, yielding 72 trajectories in $\mathcal{D}_{\log}$. Trajectories with preference returns of at least 80% of the maximum observed return form the clean endorsed set $\mathcal{D}_{\text{clean}}$. For `Coordination Ring`, we use a 60% threshold to retain a sufficient number of clean endorsements. The remaining trajectories form the background set $B = \mathcal{D}_{\log} \setminus \mathcal{D}_{\text{clean}}$.

To construct noisy feedback, we sample incorrectly endorsed trajectories $\mathcal{D}_{\text{noise}} \subseteq B$ and include them in $\mathcal{D}_{\text{like}}$. This changes the endorsement labels, not the underlying gameplay records. For each layout–style pair, we also sample a separate noise set for each value of n and reuse that set across all three training seeds. The background set B also serves as the comparison set for clean–background likelihood ranking. Table 7 reports the clean and background trajectory counts before feedback contamination.

Table 6: Preference configurations used for Styles A–C. Event weights follow the order in Tables 3, 4, and 5, respectively. c_{env} is the coefficient of the team sparse environment return.

Layout	Style	w	c_{env}
Coordination Ring	A	$(10, 10, -10, 0, -10)$	1
	B	$(10, 10, 10, 10, 0)$	0
	C	$(-10, 10, -10, 0, 0)$	1
Many Orders	A	$(-5, 5, 5, 5, 5, 0, 10, 0, 0, 10, 0)$	1
	B	$(5, 5, -5, 5, -5, 10, -10, 0, -3, -10, -1)$	1
	C	$(0, 0, 0, -5, -5, -10, -10, 3, 0, 10, -1)$	1
Distant Tomato	A	$(5, 10, 0, 0, 10, 10, 10, 10, -10, -10)$	0
	B	$(-5, 20, 0, 5, -10, -10, 10, 0, 10, 0)$	1
	C	$(-5, 0, 0, -5, 0, -10, 0, -10, 10, -10)$	0

Table 7: Trajectory counts before feedback contamination. Clean and Background denote $\mathcal{D}_{\text{clean}}$ and B, respectively. Each layout–style dataset contains $|\mathcal{D}_{\log}| = 72$ trajectories.

	Style A		Style B		Style C	
Layout	Clean	Background	Clean	Background	Clean	Background
Coordination Ring	12	60	12	60	20	52
Many Orders	16	56	14	58	6	66
Distant Tomato	12	60	10	62	6	66

C Implementation Details

C.1 Likelihood-Based Preference Reward

This part defines the auxiliary reward used in the *Task + Pref.* experiments. The reward is computed from endorsed trajectories and added to the environmental reward during RL. The AI initialization and training partners are unchanged between the task-only and reward-augmented conditions.

Frozen trajectory scorer. We train a scoring policy π_ϕ by SFT on $\mathcal{D}_{\text{like}}$ and freeze it throughout RL. It scores the AI's recorded actions using the trajectory log-likelihood $Z_\phi(\tau)$ defined in Equation 11. The scoring policy is separate from the AI policy updated during interaction. We define the empirical endorsement-score interval $I = [L_{\min}, L_{\max}]$ by

$$L_{\min} = \min_{\tau \in \mathcal{D}_{\text{like}}} Z_\phi(\tau), \qquad L_{\max} = \max_{\tau \in \mathcal{D}_{\text{like}}} Z_\phi(\tau). \tag{7}$$

These bounds use all observed endorsements, without access to the clean/noisy partition.

Trapezoidal calibration. For shoulder widths $w_-, w_+ > 0$, we define

$$\widehat{r}_{\text{pref}}(z) = \left[1 - \frac{(L_{\min} - z)_+}{w_-} - \frac{(z - L_{\max})_+}{w_+}\right]_+, \tag{8}$$

where $(x)_+ = \max(x, 0)$. The bonus equals one within I, decreases linearly on either side, and is zero beyond the two shoulders. It is bounded between zero and one; scores outside the interval receive a reduced bonus, not a negative reward. Appendix I explains the rationale and properties of this construction.

Use during RL. For an episode τ of length H, the bonus is added only at the final transition:

$$r_t^{\text{train}} = r_{\text{env}}(s_t, a_t^A, a_t^H) + \lambda\, \mathbf{1}[t = H - 1]\, \widehat{r}_{\text{pref}}(Z_\phi(\tau)), \tag{9}$$

where $\lambda \geq 0$ controls its weight. During rollout collection, we accumulate log-likelihoods under the frozen scorer and add the resulting bonus before computing returns and advantages. The scorer and calibration parameters remain fixed. Setting $\lambda = 0$ recovers the task-only condition.

Scorer and reward scale. We use the final checkpoint after 20 epochs of SFT on $\mathcal{D}_{\text{like}}$ as the frozen scorer. The plateau spans the empirical score interval $[L_{\min}, L_{\max}]$. We place the outer zero-reward endpoints at

$$L_{\text{left}} = L_{\min} - 0.05|L_{\min}|, \qquad L_{\text{right}} = L_{\max} + 0.05|L_{\max}|.$$

The shoulder widths are therefore $w_- = 0.05|L_{\min}|$ and $w_+ = 0.05|L_{\max}|$. For reward-augmented comparisons, we set $\lambda = 500$. The terminal bonus ranges from zero to 500, placing its maximum on the same order of magnitude as episodic task returns. Task-only conditions use $\lambda = 0$.

C.2 Hyperparameters for Policy Pool Construction

Our proposed framework entails the construction of two distinct policy pools. The first pool serves as the foundational starting point for the entire training pipeline, specifically designed to facilitate the population-based training required by FCP (Strouse et al., 2021).

We establish a total population size of 18 for this initial pool. To ensure sufficient behavioral diversity, these policies are systematically aggregated by sampling 3 distinct intermediate checkpoints from each of the 6 independent Self-Play (SP) training runs, initialized with different random seeds. Training against this diverse historical population ensures that the FCP agent acquires a comprehensive understanding of the environmental dynamics and develops robust, foundational task-execution capabilities. The specific hyperparameters utilized during this SP training phase are summarized in Table 8.

Furthermore, we construct the second policy pool using MAPPO with the 36 predefined preference configurations for each layout. Specifically, within each training run, we concurrently optimize a pair of distinct parameterized models: a biased policy guided by the specific preference reward, and a standard unbiased policy driven solely by the baseline environmental reward. This paired training regime ensures a strictly controlled environment for policy evaluation. The detailed hyperparameters utilized during this MAPPO training process are summarized in Table 9.

C.3 Adaptive Training for FCP, EAR, and Ours

Across all evaluated frameworks—whether it is the population-based training in FCP, the epoch-wise alternating training in EAR, or our own preference-aligned adaptation—the final stage necessitates a dedicated adaptive training phase. Fundamentally, this process governs how a cooperative agent dynamically interacts with and adapts to diverse strategies sampled from the policy pool, optimized under a specified reward formulation. To ensure a rigorous, fair, and highly reproducible comparison across all baselines and our proposed method, the hyperparameters dictating this pseudo-online learning phase are standardized. The complete configuration for this adaptation process is detailed in Table 10.

C.4 Agent Architecture

To maintain rigorous comparability with established baselines (Yu et al., 2023; Ma et al., 2026) and ensure a lightweight footprint for efficient real-time deployment, we employ a compact Convolutional Neural Network coupled with a Multi-Layer Perceptron (CNN-MLP) architecture across all evaluations.

Specifically, the spatial encoder comprises a three-layer convolutional block. These layers are sequentially configured with $\{32, 64, 32\}$ output channels. To systematically process the grid-world state inputs, every convolutional layer applies a 3×3 filter with both stride and zero-padding set to 1. Subsequently, a max-pooling operation is applied to down-sample the spatial dimensions. The resulting extracted feature embeddings are then flattened and passed through a two-layer fully connected network (MLP), which ultimately maps the representations into the final discrete action logits.

C.5 SFT and Proxy Training

We use the CNN–MLP policy architecture described in Section C.4 for all SFT models. Preference SFT starts from the FCP-trained policy π_0 and fits recorded teammate actions using the same loss

Table 8: Key hyperparameters for the self-play training via MAPPO.

Hyperparameter	**Value**
Training Setup	
Total environment steps	1×10^7
Reward shaping horizon	1×10^8
Parallel env. threads (Ring, Orders)	100
Parallel env. threads (Distant Tomato)	200
Episode length	400
Use feature normalization	True
Use reward normalization	True
Optimization	
Optimizer	Adam
Network initialization	Orthogonal
Actor learning rate	5×10^{-4}
Critic learning rate	5×10^{-4}
Optimizer epsilon	1×10^{-5}
Weight decay	0
Max gradient norm	10.0
PPO Hyperparameters	
Discount factor (γ)	0.99
GAE parameter (λ)	0.95
PPO clip parameter (ϵ)	0.2
PPO epochs	15
Number of mini-batches	1
Entropy coefficient	0.01
Value loss coefficient	1.0
Value loss function	Huber
Huber delta	10.0

form as Equation 4. The standalone SFT baselines are trained for 20 epochs on their respective clean or noisy endorsement sets, and evaluated using the final checkpoint. The likelihood scorer also uses a 20-epoch SFT checkpoint trained on $\mathcal{D}_{\text{like}}$ and remains frozen during RL.

The player proxy is trained for 20 epochs on the target player's actions in $\mathcal{D}_{\log}$. We use its final checkpoint as the frozen training partner. For both PROXY-RL and PROXY-RL+PREF., the AI instead starts from the same checkpoint after 20 epochs of preference SFT on $\mathcal{D}_{\text{like}}$. This checkpoint serves as π_{SFT} in the subsequent RL stage.

C.6 CPL

We form preference pairs between clean endorsed trajectories in $\mathcal{D}_{\text{clean}}$ and background trajectories in B. The clean trajectory is labeled as preferred in each pair. We use CPL-KL with 3,000 SFT warmup updates followed by 3,000 contrastive updates. A frozen copy of the warmup policy serves as the reference for computing the trajectory log-likelihood differences used in the contrastive objective. Across all layouts, we use batches of four preference pairs, set $\alpha = 0.1$ and the contrastive bias to 0.75, and omit the auxiliary SFT loss during contrastive training.

Table 9: Hyperparameters for the HSP MAPPO-based policy pool construction.

Hyperparameter	**Value**
Training Setup	
Total environment steps (Ring, Orders)	7.5×10^6
Total environment steps (Distant Tomato)	1×10^7
Reward shaping horizon	0
Parallel env. threads (Ring, Orders)	100
Parallel env. threads (Distant Tomato)	200
Episode length	400
Use feature normalization	True
Use reward normalization	True
Optimization	
Optimizer	Adam
Network initialization	Orthogonal
Actor learning rate	5×10^{-4}
Critic learning rate	5×10^{-4}
Optimizer epsilon	1×10^{-5}
Weight decay	0
Max gradient norm	10.0
PPO Hyperparameters	
Discount factor (γ)	0.99
GAE parameter (λ)	0.95
PPO clip parameter (ϵ)	0.2
PPO epochs	15
Number of mini-batches	1
Entropy coefficient	0.01
Value loss coefficient	1.0
Value loss function	Huber
Huber delta	10.0

C.7 Evaluation and Checkpoint Selection

Rollouts and aggregation. For each layout–style pair, we evaluate each policy with the same target-player policy over 20 stochastic episodes of 400 steps. We compute task and preference returns for each trajectory and average them within each training seed. The reported mean and standard deviation are then computed across the three seed-level averages.

Action accuracy. We evaluate action accuracy on the recorded AI state–action pairs in $\mathcal{D}_{\text{clean}}$. For each style, all methods and checkpoints use the same reference data. Accuracy is the percentage of recorded actions correctly predicted by the evaluated policy. This offline evaluation accompanies the task and preference returns measured through interaction.

Checkpoint selection. For the reported RL results, we select the checkpoint with the highest recorded training environmental return. SFT checkpoint choices are specified in Section C.5. For the stage-wise analyses, Init denotes each method's starting policy. C1–C3 are saved checkpoints chosen at approximately equal intervals in training progress before the last training checkpoint.

Table 10: Hyperparameters for the (pseudo)-online adaptive training phase (evaluated across FCP, EAR, and our method).

Hyperparameter	**Value**
Training Setup	
SFT batch size	32
SFT epoch at stage 2 (EAR)	20
Total environment steps (EAR, FCP)	2×10^7
Total environment steps (Ours)	1×10^7
Reward shaping horizon	1×10^8
Initial reward shaping factor	1.0
Parallel environment threads	200
Episode length	400
Use feature normalization	True
Use reward normalization	True
Optimization	
Optimizer	Adam
Network initialization	Orthogonal
Actor learning rate	5×10^{-4}
Critic learning rate	5×10^{-4}
Optimizer epsilon	1×10^{-5}
Weight decay	0
Max gradient norm	10.0
PPO Hyperparameters	
Discount factor (γ)	0.99
GAE parameter (λ)	0.95
PPO clip parameter (ϵ)	0.2
PPO epochs	15
Number of mini-batches	1
Entropy coefficient	0.01
Value loss coefficient	1.0
Value loss function	Huber
Huber delta	10.0

C.8 Proxy-Fitting Variants

Entropy-regularized SFT. We add an entropy term during supervised fitting of the player proxy:

$$\mathcal{L}_{\text{ent}}(\psi) = \mathbb{E}_{(s,a^H)\sim\mathcal{D}_{\log}}\left[\text{CE}\big(\hat{\mu}_\psi(\cdot \mid s), a^H\big) - 0.01\,\mathcal{H}(\hat{\mu}_\psi(\cdot \mid s))\right]. \tag{10}$$

The implementation applies cross-entropy to normalized action probabilities, so $\text{CE}(p, a) = -p_a + \log\sum_b \exp(p_b)$, and $\mathcal{H}(p) = -\sum_b p_b \log p_b$. The entropy term encourages stochastic action selection and is applied only during proxy fitting. Its coefficient remains fixed across all `Many Orders` styles and seeds. We train for 20 epochs and freeze the final checkpoint (epoch 19) as the training partner.

Checkpoint pool. We select three checkpoints from the early, middle, and final stages of a 20-epoch proxy-fitting run. These checkpoints remain frozen and form a partner pool for RL recovery. Using policies from different training stages introduces variation in the partners encountered by the AI.

Table 11: Configurations for the controlled diagnostics. λ weights the terminal preference bonus. Rollout steps count environment interactions after initialization.

AI initialization	Training partner	λ	Rollout steps
Base π_0 (Oracle)	Target user	0	1×10^7
Noisy SFT π_{SFT}	Target user	0	1×10^7
Noisy SFT π_{SFT}	Target user	500	1×10^7
Noisy SFT π_{SFT}	FCP pool	0	1×10^7
Noisy SFT π_{SFT}	FCP pool	500	1×10^7

C.9 Controlled Diagnostic Experiments

Table 11 summarizes the training conditions in Figure 3. For each style and training seed, the four SFT-initialized conditions use the same checkpoint trained on $\mathcal{D}_{\mathrm{like}}$ with $|\mathcal{D}_{\mathrm{noise}}| = 2$. The no-SFT condition starts from the corresponding FCP-trained base policy π_0. SFT checkpoint selection is described in Section C.5.

The target-user conditions use the benchmark player's policy directly, while the FCP conditions use the existing partner pool. Partner policies remain fixed during MAPPO training. For reward-augmented conditions, we use the same frozen scorer and calibration within each style and seed, with $\lambda = 500$. Task-only conditions set $\lambda = 0$. Reward construction and evaluation follow Sections C.1 and C.7, respectively.

D Extended Evaluations on Coordination Ring and Distant Tomato

In this section, we present the extended experimental results on the `Coordination Ring` and `Distant Tomato` layouts. Consistent with our primary analysis, all evaluations in this section are conducted across three independent random seeds, and we compare our proposed framework against the identical suite of baselines utilized in the `Many Orders` layout.

D.1 Results on Coordination Ring

The experimental results for the *Coordination Ring* layout are summarized in Table 12. This specific topology imposes strict spatial constraints on player movements, thereby placing extreme demands on the agent's ability to coordinate with the user's stylistic actions. Due to the inherently limited event space in this layout, the dataset quality is relatively constrained. Applying the standard 80% maximum Preference Score threshold to construct the positive dataset yielded an impractically sparse distribution (e.g., retrieving only 2 viable trajectories for certain styles). Consequently, we relaxed the commendation threshold to 60% to ensure a functional $\mathcal{D}_{clean}$ subset. Under these relaxed conditions, the internal consistency of the annotated offline trajectories inevitably decreases. We observe that standard SFT approaches suffer from severe instability and sub-optimal mean performance, a degradation that is catastrophically amplified when label noise is introduced. In stark contrast, Proxy-RL and it's variants successfully navigate this noisy, constrained environment, consistently achieving state-of-the-art or near-optimal preference alignment across all three tested styles.

D.2 Results on Distant Tomato

Table 13 compares SFT and Proxy-RL at $n = 1, \ldots, 4$. In Style A, SFT preference scores vary substantially across the noisy datasets, while Proxy-RL stays within 2340–2430, close to the clean-reference score of 2340. Its task returns are similarly consistent across the four settings. Although clean SFT achieves higher scores, $n = 2$ is the only noisy condition in which it retains that level of performance. Proxy-RL instead maintains strong performance across all four noisy-feedback settings. In Style B, Proxy-RL obtains broadly comparable preference scores and higher mean task returns than SFT at each noise count. These results extend its effectiveness beyond the $n = 4$ condition.

Style C presents a conflict between the specified preference and task progress. The clean reference achieves a preference score of 110 with a task return of only 10. The preferred reference behavior therefore receives little task reward, so task-oriented recovery can pull the policy away from this style. Across all four noise settings, Proxy-RL achieves higher task returns but lower preference scores than SFT. The preference bonus provides additional guidance toward the specified style and partially restores preference performance. At $n = 4$, it improves the preference score from -341 to -111, while task return decreases from 257 to 144. The bonus thus shifts the balance toward the preferred behavior rather than improving both objectives together.

Table 12: Performance across different preference styles. Each method is evaluated using Preference Score (↑) and Environment Return (↑). All reported mean and standard deviation values are rounded to three significant figures. Results are reported on the `Coordination Ring` layout.

Method	**Style A**		**Style B**		**Style C**	
	Pref. Score	Env. Return	Pref. Score	Env. Return	Pref. Score	Env. Return
Offline Trajectories	527	167	475	165	240	115
ZSC	169 ± 18.2	84.0 ± 7.26	338 ± 31.6	84.0 ± 7.26	32.2 ± 14.3	110 ± 19.9
Oracle	253 ± 2.09	131 ± 7.54	503 ± 3.30	131 ± 7.54	65.5 ± 7.18	$\mathbf{161 \pm 1.70}$
EAR	281 ± 6.33	132 ± 10.2	278 ± 138	72.0 ± 48.0	20.3 ± 6.94	67.7 ± 40.3
SFT	406 ± 94.9	121 ± 34.9	466 ± 38.5	148 ± 14.4	126 ± 11.3	45.7 ± 9.57
SFT ($n = 1$)	352 ± 74.3	102 ± 23.4	479 ± 119	153 ± 40.7	177 ± 13.9	72.7 ± 7.41
SFT ($n = 2$)	335 ± 30.3	96.3 ± 7.76	448 ± 41.8	147 ± 16.7	110 ± 77.9	43.0 ± 30.6
CPL	122 ± 94.1	31.3 ± 30.6	171 ± 17.6	42.3 ± 3.09	125 ± 50.0	64.7 ± 28.9
PROXY-RL ($n = 2$)	587 ± 117	190 ± 42.7	556 ± 41.5	185 ± 16.9	172 ± 24.8	127 ± 6.13
with entropy-regularized SFT	581 ± 31.1	187 ± 11.6	546 ± 36.2	188 ± 12.3	$\mathbf{201 \pm 19.6}$	131 ± 3.09
with checkpoint pool ($K = 3$)	$\mathbf{620 \pm 80.3}$	$\mathbf{205 \pm 27.8}$	$\mathbf{581 \pm 14.7}$	$\mathbf{196 \pm 12.0}$	169 ± 45.4	129 ± 3.30
PROXY-RL+PREF. ($n = 2$)	551 ± 141	178 ± 49.2	553 ± 60.6	184 ± 24.2	186 ± 52.2	121 ± 9.84

Table 13: Performance across different preference styles. Each method is evaluated using Preference Score (↑) and Environment Return (↑). All reported mean and standard deviation values are rounded to three significant figures. Results are reported on the `Distant Tomato` layout.

Method	**Style A**		**Style B**		**Style C**	
	Pref. Score	Env. Return	Pref. Score	Env. Return	Pref. Score	Env. Return
Offline Trajectories	2340	412	621	266	110	10.0
ZSC	1040 ± 28.5	194 ± 4.71	-502 ± 13.2	226 ± 2.36	-880 ± 4.83	228 ± 3.30
Oracle	1070 ± 13.7	211 ± 3.09	-472 ± 6.77	225 ± 6.38	-869 ± 25.3	230 ± 0.816
EAR	1980 ± 682	353 ± 115	-163 ± 472	225 ± 5.89	-879 ± 30.1	227 ± 7.32
SFT	$\mathbf{2500 \pm 1.53}$	$\mathbf{440 \pm 0.471}$	$\mathbf{731 \pm 9.46}$	$\mathbf{365 \pm 9.81}$	$\mathbf{97.2 \pm 4.64}$	16.3 ± 10.5
SFT ($n = 1$)	447 ± 124	58.0 ± 17.7	636 ± 65.8	329 ± 34.1	72.9 ± 16.9	23.7 ± 12.7
SFT ($n = 2$)	2480 ± 26.2	439 ± 4.19	650 ± 45.3	303 ± 12.2	-25.7 ± 45.6	72.7 ± 18.9
SFT ($n = 3$)	1040 ± 749	170 ± 139	631 ± 72.2	319 ± 41.4	54.0 ± 18.7	24.7 ± 9.74
SFT ($n = 4$)	492 ± 298	75.3 ± 53.5	628 ± 98.1	329 ± 41.8	-52.2 ± 12.2	81.7 ± 15.5
CPL	0.00 ± 0.00	0.00 ± 0.00	248 ± 23.8	166 ± 11.0	49.3 ± 51.8	69.0 ± 48.6
PROXY-RL ($n = 1$)	2430 ± 29.5	427 ± 4.99	650 ± 21.0	361 ± 14.8	-415 ± 237	257 ± 4.24
PROXY-RL ($n = 2$)	2370 ± 48.7	417 ± 8.65	590 ± 39.7	312 ± 18.0	-499 ± 190	236 ± 26.2
PROXY-RL ($n = 3$)	2400 ± 31.9	425 ± 6.34	605 ± 113	335 ± 61.6	-476 ± 127	241 ± 16.0
PROXY-RL ($n = 4$)	2340 ± 24.8	412 ± 3.86	648 ± 43.0	342 ± 11.5	-341 ± 195	257 ± 6.16
with entropy-regularized SFT	2370 ± 19.5	419 ± 3.09	618 ± 48.2	340 ± 21.7	-325 ± 203	266 ± 8.65
with checkpoint pool ($K = 3$)	2330 ± 92.0	416 ± 12.7	605 ± 25.4	335 ± 14.6	-161 ± 16.5	$\mathbf{267} \pm 10.8$
PROXY-RL+PREF. ($n = 4$)	2300 ± 29.4	418 ± 2.36	664 ± 19.6	360 ± 12.0	-111 ± 26.7	144 ± 44.3

E COMPUTATIONAL RESOURCES

All empirical evaluations and model training were conducted on a single workstation equipped with one NVIDIA H100 Tensor Core GPU. Thanks to the memory efficiency of our implementation, a single H100 GPU can seamlessly support 3 to 4 concurrent training processes. Specifically, each independent training run, configured with 200 parallel environment rollout threads (`n_rollout_threads = 200`), stably consumes less than 20 GB of VRAM.

To provide a transparent overview of the computational overhead, Table 15 details the wall-clock time required to execute a single, isolated training run. We specifically benchmark the two primary training

Table 14: Average number of deliveries performed by the cooperative agent per episode on the *Many Orders* layout, averaged across three training seeds. Offline trajectories provide a reference for the preferred teammate behavior. Both proxy-based variants use a single learned user proxy ($K = 1$) and reproduce the zero-delivery pattern in Styles A and B, with delivery counts close to the offline reference in Style C.

Method	**Style A**	**Style B**	**Style C**
Offline Trajectories	0	0	11.2
Oracle	9.03	9.23	14.6
EAR	2.22	2.08	11.0
PROXY-RL	**0**	**0**	11.9
PROXY-RL+PREF.	**0**	**0**	11.4

phases, SFT and the Pseudo-Online MAPPO, using the `Many Orders` layout as the representative testbed.

Table 15: Wall-clock time for a single training run on the `Many Orders` (Style A) layout.

Training Phase	**Algorithm**	**Wall-Clock Time**
Stage 1 & 2	SFT (20 Epochs)	148s
Stage 3	Pseudo-Online: Env. Rollout (per iteration)	15.2s
Stage 3	Pseudo-Online: PPO Update (15 epochs)	20.3s

F LIMITATIONS AND FUTURE WORK

Our evaluation focuses on three Overcooked layouts that cover complementary coordination challenges, including spatial constraints, task allocation, and role-dependent collaboration. While this setting provides a sufficiently rich and controlled testbed for studying the adaptation mechanisms considered in this work, we have not yet established whether the same findings carry over to high-fidelity, large-scale commercial games, where interaction dynamics and behavioral spaces can be substantially more complex. In addition, our benchmark relies on simulated players and synthetically specified preference styles. Future work will investigate broader and more human-like behavioral distributions, preference patterns, and feedback conditions to better assess the generality of the observed phenomena.

G PREFERENCE RECOGNITION AND EXECUTION UNDER LOCALIZED NOISE

This section explains how a policy can retain a preference-related ranking of recorded trajectories while becoming less reliable during execution. We analyze the effect of conflicting action labels in a tabular SFT model, then show how the same noisy trajectories contribute to pairwise ranking accuracy.

Setup. Let $C = \mathcal{D}_{\text{clean}}$ and $N = \mathcal{D}_{\text{noise}}$ denote disjoint sets of correctly and incorrectly endorsed trajectories, with $\mathcal{D}_{\text{like}} = C \dot{\cup} N$. The background set is $B = \mathcal{D}_{\log} \setminus C$. Pairwise evaluation compares each trajectory in C with every trajectory in B; comparisons within either set are excluded. Since $N \subseteq B$, we write $B = G \dot{\cup} N$, where $G = B \setminus N$. Let $m = |C|$, $n = |N|$, and $M = |B|$. For the noisy case, assume $m > 0$ and $0 < n < M$.

A frozen policy π_ϕ scores the AI's recorded actions by

$$\ell_\phi(s, a^A) = \log \pi_\phi(a^A \mid s), \qquad Z_\phi(\tau) = \sum_{t=0}^{H-1} \ell_\phi(s_t, a_t^A), \tag{11}$$

where H is the episode length. The first result uses an idealized tabular policy. The second holds for any fixed trajectory score.

G.1 Conflicting Labels and Sequential Execution

The fraction of incorrectly endorsed trajectories does not directly specify the action error at a particular state. What matters for that decision is the fraction of training labels that conflict with the required action. To illustrate how local fitting errors accumulate during execution, we analyze an idealized tabular model trained by negative-log-likelihood minimization.

Proposition G.1 (Sequential amplification of conflicting labels)**.** Consider an idealized tabular SFT model trained by unregularized negative-log-likelihood minimization, with an unrestricted action distribution at each state. A specified preferred behavior requires K ordered decisions: action a_i^+ at state s_i for $i = 1, \dots, K$. Assume that the first decision occurs at s_1 and, whenever all earlier required actions have been selected, the next required decision occurs at s_i before the episode ends.

At s_i, let $c_i > 0$ be the number of training labels equal to a_i^+ and let n_i count all other labels. Define the local conflict rate

$$\eta_i = \frac{n_i}{c_i + n_i}.$$

The SFT optimum satisfies

$$\pi_{\mathrm{BC}}(a_i^+ \mid s_i) = 1 - \eta_i.$$

Under stochastic execution, let E_K denote the event that all K required actions are selected. Then

$$\Pr(E_K) = \prod_{i=1}^{K} (1 - \eta_i).$$

If $\eta_i \geq \eta$ for every i, the probability of at least one deviation from this sequence satisfies

$$\Pr(E_K^c) \geq 1 - (1 - \eta)^K.$$

Proof. At each state, the cross-entropy minimum is attained by assigning actions their empirical conditional frequencies. Thus, $\pi_{\mathrm{BC}}(a_i^+ \mid s_i) = c_i/(c_i + n_i)$.

Let E_i denote correct completion of the first i required decisions, and let E_0 be the certain event. By the reachability assumption, conditional on E_{i-1}, the next required decision occurs at s_i. Hence

$$\Pr(E_i \mid E_{i-1}) = 1 - \eta_i.$$

Applying the chain rule gives

$$\Pr(E_K) = \prod_{i=1}^{K} \Pr(E_i \mid E_{i-1}) = \prod_{i=1}^{K} (1 - \eta_i).$$

The final bound follows from $1 - \eta_i \leq 1 - \eta$. □

A noisy trajectory can introduce conflicting labels at several required decisions. The proposition shows how these local conflicts can reduce the probability of completing the entire sequence. It does not assume that errors at different stages are independent. The result describes stochastic execution in the tabular model; it is not a general bound on the task or preference return of a neural policy.

G.2 Noise Weight in Pairwise Ranking

The fraction of incorrectly endorsed trajectories in the SFT dataset is

$$\epsilon = \frac{n}{m + n}. \tag{12}$$

With equal-length trajectories and uniform weighting, ϵ is also the fraction of training action samples drawn from N. It is not necessarily the fraction of incorrect actions: a mislabeled trajectory may contain both matching and conflicting actions.

For pairwise evaluation against the background set, the fraction of comparisons involving N is

$$\rho = \frac{n}{M} = \frac{\epsilon}{1-\epsilon}\frac{m}{M}. \tag{13}$$

The same noisy trajectories can therefore represent a substantial fraction of a small SFT dataset but only a small fraction of comparisons against a large background set.

For $X \in \{B, G, N\}$, define

$$\widehat{A}_{C,X} = \frac{1}{m|X|} \sum_{c \in C} \sum_{x \in X} \mathbf{1}[Z_\phi(c) > Z_\phi(x)]\,. \tag{14}$$

This definition uses all clean–background pairs; ties do not count as correct rankings.

Proposition G.2 (Exact pairwise decomposition)**.** For any fixed score function,

$$\widehat{A}_{C,B} = (1-\rho)\widehat{A}_{C,G} + \rho\widehat{A}_{C,N}. \tag{15}$$

If $\widehat{A}_{C,G} \geq 1-\delta$, with $\delta \in [0,1]$, then

$$\widehat{A}_{C,B} \geq (1-\rho)(1-\delta) \geq 1-\rho-\delta. \tag{16}$$

Proof. Since $B = G \dot\cup N$, the mM comparisons split into $m(M-n)$ clean–generic pairs and mn clean–noise pairs. Their respective proportions are $1-\rho$ and ρ, which gives the identity. The lower bound follows from $\widehat{A}_{C,N} \geq 0$ and $\widehat{A}_{C,G} \geq 1-\delta$. □

When ρ is small, overall accuracy can remain high even if clean trajectories are poorly separated from incorrectly endorsed ones. This requires the clean–generic ranking to remain accurate. The decomposition explains the weight of each type of comparison; it does not guarantee that noisy training preserves this ranking. With no incorrectly endorsed trajectories, $B = G$ and $\widehat{A}_{C,B} = \widehat{A}_{C,G}$.

Example: Many Orders, Style C. With $m = 6$ clean trajectories, $n = 2$ incorrectly endorsed trajectories, and $M = 66$ background trajectories, the two proportions are

$$\epsilon = \frac{2}{6+2} = 25\%, \qquad \rho = \frac{2}{66} \approx 3.03\%.$$

The incorrectly endorsed trajectories account for one quarter of the SFT data, but only about three percent of the clean–background comparisons. If clean trajectories remain well separated from the remaining background, overall ranking accuracy can therefore stay high even when the model fails to distinguish clean examples from incorrectly endorsed ones.

Interpretation. High pairwise accuracy means that, averaged over clean trajectories, a large fraction of background trajectories receive lower scores. The high accuracy observed in Figure 2 therefore shows that the SFT policy retains preference-related ranking information on the evaluated data. This information is present in the policy's own likelihood scores, without an additional classifier.

Reliable execution requires the policy to select appropriate actions throughout an interaction. The first result shows how local action conflicts can reduce the reliability of a required sequence. The second shows why ranking errors involving a small background subset may have little effect on overall accuracy. Together, they explain how retained preference recognition can coexist with unreliable execution.

H Action Accuracy and Cooperative Behavior during Adaptation

We evaluate recorded checkpoints of Proxy-RL, EAR, and Oracle in `Many Orders`. Within each style, all policies are evaluated with the same target player. Init denotes each method's own starting policy, and C1–C3 denote intermediate checkpoints. Selected is the checkpoint chosen according to the evaluation protocol in Section C.7.



Figure 6: Action accuracy and rollout returns at recorded checkpoints in `Many Orders`. Columns correspond to preference styles. Dashed lines show clean offline-reference returns. Selected denotes the checkpoint used for evaluation.

Action accuracy and rollout performance. Figure 6 tracks clean-set action accuracy alongside preference and task returns. Proxy-RL improves both mean returns from its SFT initialization across all three styles, without corresponding gains in action accuracy. In Style A, Proxy-RL maintains nearly constant accuracy as both returns improve, whereas EAR loses accuracy and rollout performance relative to its own initialization. In Style B, EAR and Proxy-RL maintain similar action accuracies, yet Proxy-RL achieves higher preference returns at later checkpoints. In Style C, Proxy-RL's action accuracy decreases slightly while its preference return increases. These results distinguish agreement with recorded actions from the ability to sustain preferred behavior during cooperation. Action accuracy alone is therefore insufficient to track preference retention and recovery during RL.

Changes in task allocation. Figure 7 tracks AI event frequencies per 100 environment steps to identify the behavioral changes accompanying adaptation. In Styles A and B, Proxy-RL leaves soup delivery to the player, matching the clean reference trajectories. During recovery, it increases tomato pickups in Style A and *Optimal placement* events in Style B. These changes accompany higher task returns (Figure 6). EAR instead begins delivering soup at later checkpoints, adding a responsibility that the reference trajectories leave to the player. Table 14 provides the per-episode delivery counts and extends the comparison to PROXY-RL+PREF., which also remains close to the reference delivery pattern.

Style C shows why delivery counts alone do not fully describe the change in cooperation. EAR and Proxy-RL have similar soup-delivery frequencies, but EAR increasingly performs *Optimal placement* events. These events remain nearly absent from both the reference trajectories and Proxy-RL. Across these comparisons, EAR takes on tasks beyond those performed by the AI in the reference trajectories. Proxy-RL instead improves task performance while remaining closer to the reference allocation of work. These event-level results show how similar action accuracy can mask differences in the collaboration actually performed.

I THEORETICAL MOTIVATION FOR REWARD CALIBRATION

The preceding analysis in section G distinguishes trajectory recognition from reliable execution. Here we explain how we convert the frozen trajectory score into an auxiliary reward. The construction uses a flat region because positive labels do not specify a preference ranking within the endorsed set. It also limits incentives to increase likelihood beyond the observed score range.

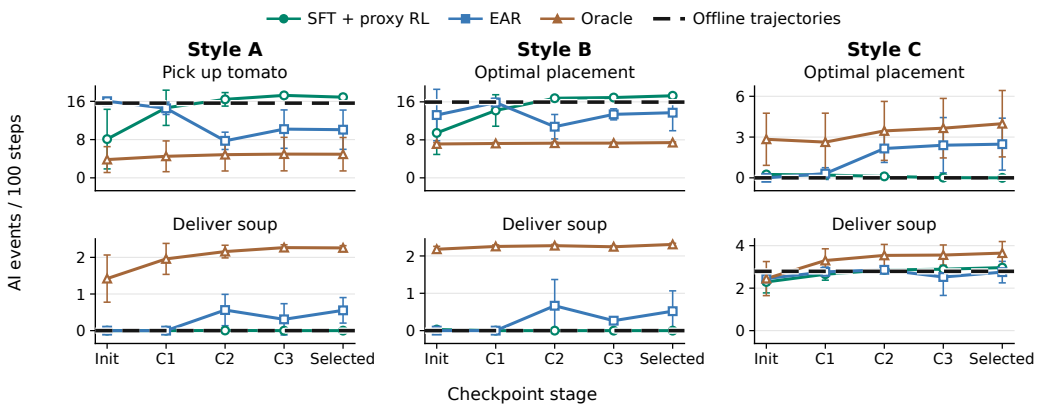

Figure 7: AI event frequencies at the same checkpoints, measured per 100 environment steps. Top: tomato pickups in Style A and *Optimal placement* in Styles B and C. Bottom: soup deliveries. Dashed lines show clean offline-reference frequencies.

I.1 What Positive Labels Identify

Lemma I.1 (No ordering from positive labels alone)**.** Suppose a positive label indicates only that a trajectory's latent utility $u(\tau)$ exceeds a threshold b, with no further restrictions on u. Positive labels alone do not identify the utility ordering of the endorsed trajectories.

Proof. Consider two endorsed trajectories and assign them distinct utilities above b. Swapping these utilities reverses their ordering without changing either label. Both orderings are therefore consistent with the observed positive labels. □

Even with noiseless endorsements, positive labels provide no ordering among the accepted trajectories. We therefore assign the same bonus throughout the observed endorsement-score interval I. The plateau rewards membership in this interval without introducing a likelihood-based ranking within it.

I.2 Local Effects of Rewarding Raw Likelihood

Fix a state s and let $\nu(\cdot \mid s)$ be a distribution over its available actions. The expected raw log-likelihood score is

$$R_s(\nu) = \sum_a \nu(a \mid s)\ell_\phi(s, a), \tag{17}$$

where ℓ_ϕ is defined in Equation 11.

Proposition I.2 (Statewise mode preference)**.** Assume a finite available-action set and finite scores $\ell_\phi(s, a)$. For every action distribution $\nu(\cdot \mid s)$,

$$R_s(\nu) \le \max_a \ell_\phi(s, a). \tag{18}$$

Equality holds if and only if ν assigns probability only to actions attaining the maximum score.

Proof. Let $M_s = \max_a \ell_\phi(s, a)$. Then

$$M_s - R_s(\nu) = \sum_a \nu(a \mid s)\big(M_s - \ell_\phi(s, a)\big) \ge 0.$$

Equality holds exactly when every action with positive probability has score M_s. □

At a fixed state, the local objective favors the scoring policy's highest-probability actions. This includes repeated or inactive actions when they receive the highest scores. We therefore use a bounded score interval as the reward target, instead of rewarding every increase in raw likelihood.

I.3 A PIECEWISE-LINEAR BAND REWARD

We replace the hard band indicator $\mathbf{1}[z \in I]$ with the continuous, piecewise-linear reward in Equation 8. Its linear shoulders provide graded bonuses near the interval boundaries.

Proposition I.3 (Piecewise-linear band reward)**.** For fixed $L_{\min} \leq L_{\max}$ and $w_-, w_+ > 0$, Equation 8 is the unique continuous function that:

1. equals one on $I = [L_{\min}, L_{\max}]$;
2. is affine on $[L_{\min} - w_-, L_{\min}]$ and $[L_{\max}, L_{\max} + w_+]$;
3. equals zero outside $[L_{\min} - w_-, L_{\max} + w_+]$.

Its slopes on the interiors of the left shoulder, plateau, and right shoulder are $1/w_-$, 0, and $-1/w_+$, respectively. Its maximizer set is exactly I.

Proof. Continuity fixes the values at the outer endpoints to zero and the values at both ends of I to one. Each shoulder is therefore the unique affine interpolation between its endpoint values. The function is fixed on the plateau and outside the shoulders. The slopes and maximizer set follow directly. □

The left shoulder increases the bonus as scores approach the endorsement interval from below. The plateau assigns the maximum bonus throughout the interval, while the right shoulder reduces the bonus for scores above it. Together, these components favor scores within the observed range and limit incentives for further likelihood increases.

The resulting reward is bounded and calibrated using the observed endorsements. The propositions establish the local incentive of raw likelihood and the properties of the piecewise-linear calibration. The main-text experiments evaluate how this auxiliary reward affects task performance and preference satisfaction.