CROSS-ENVIRONMENT COOPERATION WITH DECOUPLED POLICY AND VALUE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Zero-shot coordination (ZSC) enables ego agents to collaborate with unseen partners and is essential for human–AI coordination with diverse human behaviors. Recent work has improved coordination generalization by training agents across diverse environments. However, cross-environment training can still be limited by overfitting. Prior work in single-agent reinforcement learning shows that shared policy-value representations can hinder generalization by introducing environment-specific information into the policy representation. We hypothesize that shared policy-value representations also limit generalization in cross-environment ZSC. To investigate this hypothesis, we incorporate IDAAC, which decouples policy and value learning, into Cross-Environment Cooperation and introduce DCEC. Our analysis shows that DCEC selects actions more consistently across similar states from different environments, while encoding less environment-specific information in its policy representation. Moreover, DCEC exhibits higher value-gradient GSNR than CEC at larger training batch sizes, indicating more consistent value-learning gradients under larger-batch training. Across Dual Destination and Overcooked-AI, DCEC improves coordination with unseen partners and generalization to unseen environments, while also achieving strong coordination performance with real humans.

1 INTRODUCTION

In real-world cooperative tasks, agents must coordinate with partners whose behaviors are unknown during training, such as human drivers in autonomous driving, users interacting with assistive robots, or human teammates in cooperative games. Achieving success in these settings requires agents to learn general coordination strategies that are robust across diverse and suboptimal partners, including humans. Since it is impractical to train agents with every possible partner they may encounter after deployment, they must generalize beyond the partner behaviors observed during training. To address this challenge, Zero-Shot Coordination (ZSC) aims to develop ego agents that can effectively collaborate with previously unseen partners.

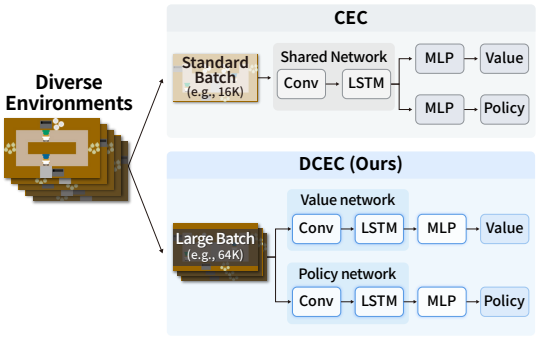


Figure 1: **Overview of DCEC.** CEC uses a shared policy–value network with a standard training batch, whereas DCEC separates the policy and value networks and benefits from larger-batch cross-environment training.

A straightforward way to improve generalization to unseen partners is increasing the diversity of training experiences. Early approaches in ZSC improved partner diversity through population-based training methods (Strouse et al., 2021; Lupu et al., 2021; Zhao et al., 2023; Yu et al., 2023; Sarkar et al., 2023). These methods expose ego agents to various simulated partners and encourage them to learn coordination strategies that are robust to different conventions and behaviors. The research focus later expanded to environment diversity, as coordination strategies also depend on environment

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.

structure. This motivated approaches that diversify the environments, rather than varying only the partners within a fixed environment. Environment generation methods such as Unsupervised Environment Design (UED) (Dennis et al., 2020) have been adopted in ZSC to expose agents to diverse environment structures (Ruhdorfer et al., 2024; 2025; You et al., 2025). Surprisingly, a recent work showed that environment diversity alone is sufficient to bring significant generalization benefit on both unseen partners and unseen environments (Jha et al., 2025).

However, increasing data diversity alone does not guarantee generalization. Even when agents are trained on diverse environments, they still overfit to features that appear during training but don't transfer to unseen tasks or partners. In single-agent RL, many approaches have been proposed to mitigate overfitting, such as regularization, data augmentation, and improved training data sampling, because agents tend to overfit to early training data and show poor generalization performance. In particular, shared policy-value architectures lead the policy to encode environment-specific features for value estimation, thereby hindering generalization (Raileanu & Fergus, 2021). Nevertheless, prior work (Lupu et al., 2021; Zhao et al., 2023; Jha et al., 2025) on ZSC has primarily focused on data diversity (e.g., partner or environment diversity), while the role of overfitting in ZSC generalization remains largely unexplored.

Motivated by this gap, we investigate whether overfitting is a critical bottleneck in ZSC. We hypothesize that overfitting significantly limits cooperative agents' generalization, resulting in poor ZSC. Our experiments use the Cross-Environment Cooperation (CEC) framework, a state-of-the-art ZSC method that leverages environment diversity. We find evidence that overfitting limits the generalization benefit of cross-environment training, and investigate shared policy–value representations as a potential contributor to this overfitting, motivated by prior observations in single-agent RL.

To address this problem, we integrate IDAAC (Raileanu & Fergus, 2021), which decouples policy and value learning and introduces auxiliary losses for policy generalization, into the Cross-Environment Cooperation framework. We call this approach DCEC. DCEC significantly improves ZSC performance over vanilla CEC. Our analysis further shows that DCEC learns policy representations with less environment-specific information and makes more consistent decisions across similar states from different environments. These results support the hypothesis that shared representations contribute to cross-environment overfitting. In addition, we find that DCEC benefits from larger training batches, whereas vanilla CEC shows no consistent improvement as the batch size increases from 8,192 to 65,536.

We evaluate DCEC on the Dual Destination problem and Overcooked-AI across five standard tasks and procedurally generated environments, using cross-play with independently trained agents from different random seeds. In Overcooked-AI, we further evaluate cross-play with agents trained using different algorithms and human-AI coordination with real human participants, and find that our approach achieves stronger overall performance than the CEC baseline.

2 Related Work

2.1 Zero-Shot Coordination

Zero-Shot Coordination (ZSC) trains an ego agent that effectively collaborates with partners not encountered during training. Early ZSC research (Strouse et al., 2021) mainly focused on increasing training-partner diversity through the Population-Based Training (PBT) method. These approaches expose an ego agent to pre-trained partners with different behaviors and coordination strategies from the population. This encourages the ego agent to learn a policy that is robust to diverse partners. Following this method, many works have been proposed to improve population diversity using statistical diversity measures (Charakorn et al., 2023; Lupu et al., 2021; Zhao et al., 2023), reward shaping (Yu et al., 2023), cross-play performance (Sarkar et al., 2023; Wang et al., 2025; Loo et al., 2025), and generative models (Liang et al., 2024; Chaudhary et al., 2025; Li et al., 2026). Although these methods use different techniques, they share the same underlying goal, which is improving generalization to unseen partners by increasing partner diversity during training.

More recent work has expanded ZSC from partner diversity to environment diversity. Coordination strategies depend not only on partner behavior but also on the structure of the environment. Environment diversity has been shown to improve coordination ability in ZSC (McKee et al., 2022).

Unsupervised Environment Design (UED) based methods have also been applied to ZSC to automatically generate diverse training environments for cooperative agents (Ruhdorfer et al., 2024; You et al., 2025; Ruhdorfer et al., 2025). Cross-Environment Cooperation (CEC) (Jha et al., 2025) uses self-play across diverse environments to improve generalization to both unseen partners and unseen environments. Overall, prior work has demonstrated the importance of both partner and environment diversity for ZSC. In contrast, we focus on how policy-value learning affects generalization when agents are trained across diverse environments.

2.2 Generalization in Deep Reinforcement Learning

Deep reinforcement learning suffers from overfitting, which limits generalization to unseen environments. To address this issue, prior work has adopted regularization techniques from supervised learning, such as dropout (Hiraoka et al., 2021; Igl et al., 2019), weight decay (Farebrother et al., 2018), batch normalization (Igl et al., 2019), and data augmentation (Cobbe et al., 2019; Laskin et al., 2020; Raileanu et al., 2020; Wang et al., 2020). Other approaches improve generalization by reducing transient non-stationarity through iterative policy distillation (Igl et al., 2020) or by prioritizing training levels with high learning potential during sampling (Jiang et al., 2021).

Another line of work has studied the effect of actor-critic architectures on generalization. Prior work (Raileanu & Fergus, 2021) has proposed decoupling the actor and critic networks to reduce interference between policy and value learning in shared architectures. Environment-specific features required for value estimation can otherwise influence the policy representation and lead to poor generalization. The other study (Moon et al., 2022) has shown that value learning itself can overfit to training data in shared actor-critic architectures. They proposed that controlling the value function's update frequency and learning schedule improves generalization performance. In this work, we apply IDAAC to zero-shot coordination under cross-environment training and examine its effect on generalization to unseen partners and environments. We further investigate whether this architecture benefits from larger training batches.

3 Preliminaries

3.1 Cooperative Markov Games

We consider a finite-horizon fully cooperative Markov game with two agents that share a common reward. A cooperative Markov game is defined as $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, R, H \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space of each agent, $\mathcal{T}$ is the transition function, R is the shared reward function, and H is the episode horizon. At each timestep t, the two agents choose actions $a_t^1, a_t^2 \in \mathcal{A}$, and the next state is sampled from the transition dynamics $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t^1, a_t^2)$. Both agents receive the same reward $R(s_t, a_t^1, a_t^2)$.

In our setting, each task $m \sim \mathcal{M}$ corresponds to a particular environment configuration. The layout determines the initial state distribution $p(s_0 \mid m)$, while the reward function R and transition function T are shared across tasks. For example, in Overcooked, different tasks may correspond to different layouts and object configurations. Given a task m, we define the cooperative performance of a cooperator policy π_C and a partner policy π_p as $S(\pi_p, \pi_C, m) = \mathbb{E}\left[\sum_{t=0}^{H} R(s_t, a_t^p, a_t^C)\right]$, where the expectation is taken over trajectories induced by $s_0 \sim p(s_0 \mid m)$, $a_t^p \sim \pi_p(\cdot \mid o_t^p)$, $a_t^C \sim \pi_C(\cdot \mid o_t^C)$, and $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t^p, a_t^C)$.

3.2 Cross-Environment Cooperation

Self-play optimizes a policy on a single task by interacting with a copy of itself. Given a task m_0, the self-play objective is defined as $J_{\mathrm{SP}}(\pi_C) = S(\pi_C, \pi_C, m_0)$. Although this objective can produce strong performance on the training task, it does not train the policy to generalize across tasks or partners

Population-based training (Strouse et al., 2021; Zhao et al., 2023; Yu et al., 2023) addresses partner generalization by training the cooperator against a population of simulated partner policies $\mathcal{P} = \{\pi_1, \pi_2, ..., \pi_n\}$ on a fixed task. Its objective is $J_{\mathrm{PBT}}(\pi_C) = \mathrm{E}_{\pi_i \sim \mathcal{P}}\left[S(\pi_i, \pi_C, m_0)\right]$. This objective exposes the cooperator to diverse partner behaviors, but the environment task remains fixed.

Cross-Environment Cooperation(CEC) (Jha et al., 2025) considers a distribution of cooperative tasks. Let $m \sim \mathcal{M}$ denote a task sampled from the task distribution. The cross-environment cooperation objective trains a single cooperator policy through self-play across diverse tasks using the objective $J_{\mathrm{CEC}}(\pi_C) = \mathrm{E}_{m_i \sim \mathcal{M}}[S(\pi_C, \pi_C, m_i)]$. By optimizing performance over diverse tasks, this objective encourages the policy to learn cooperative behaviors that transfer across diverse environment structures. Since different environments require different coordination strategies, training across them can promote more general cooperative skills that transfer not only to unseen tasks but also to unseen partners. CEC uses a shared representation for the policy and value functions. In practice, CEC collects trajectories from multiple tasks in parallel environments, so each training update is based on sampled tasks. In this work, we use the same cross-environment objective and compare different policy-value learning formulations within this framework.

4 DCEC: Decoupled actor-critic for Cross-Environment Cooperation

In this section, we examine our policy-value decoupling hypothesis using DCEC, which incorporates IDAAC (Raileanu & Fergus, 2021) into CEC. CEC and DCEC use the same cross-environment objective but differ in their policy-value learning.

Shared Policy-Value Learning. CEC uses a shared representation for both the policy and value functions. As a result, environment-specific features learned for value estimation can also influence the policy representation, potentially limiting cross-environment generalization.

Decoupled Policy-Value Learning. DCEC follows the IDAAC formulation, using separate policy and value networks together with auxiliary objectives for policy generalization. For agent $i \in \{1, 2\}$ receiving observation o_t^i under task m, the representations are computed separately as

$$(z_t^{\pi,i}, h_t^{\pi,i}) = f_\theta^\pi(z_{t-1}^{\pi,i}, o_t^i), \quad (z_t^{V,i}, h_t^{V,i}) = f_\phi^V(z_{t-1}^{V,i}, o_t^i), \tag{1}$$

where θ and ϕ denote the parameters of the policy and value networks, respectively. The variables $z_t^{\pi,i}$ and $z_t^{V,i}$ denote the recurrent states of the policy and value networks, while $h_t^{\pi,i}$ and $h_t^{V,i}$ denote the corresponding output representations. The policy and value functions are then computed as

$$\pi_\theta(a_t^i \mid o_t^i) = \pi_\theta(a_t^i \mid h_t^{\pi,i}), \quad V_\phi(o_t^i) = V_\phi(h_t^{V,i}). \tag{2}$$

With separate policy and value networks, value-loss gradients update only the value-network parameters ϕ and do not propagate through the policy network. The value network is optimized over trajectories collected by both agents using the mean squared value error:

$$\mathcal{L}_V(\phi) = \mathbb{E}_{m \sim \mathcal{M}}\left[\mathbb{E}_{\tau_m \sim (\pi_\theta, \pi_\theta)}\left[\frac{1}{2H}\sum_{t=0}^{H}\sum_{i=1}^{2}\left(V_\phi(o_t^i) - \hat{V}_t^i\right)^2\right]\right], \tag{3}$$

where $\hat{V}_t^i$ denotes the value target computed by Generalized Advantage Estimation (GAE) using the shared reward $R(s_t, a_t^1, a_t^2)$.

The policy network is optimized with PPO under the CEC self-play objective. Following IDAAC, we additionally use advantage-prediction $\mathcal{L}_A$ and instance-invariance $\mathcal{L}_E$ objectives for policy generalization:

$$\mathcal{J}_\pi(\theta) = \mathcal{J}_{\mathrm{PPO}}(\theta) + \alpha_s \mathcal{S}_\pi(\theta) - \alpha_a \mathcal{L}_A(\theta) - \alpha_i \mathcal{L}_E(\theta), \tag{4}$$

where $\mathcal{J}_{\mathrm{PPO}}$ is the PPO policy objective and $\mathcal{S}_\pi$ is the entropy bonus. Detailed formulations of the auxiliary objectives and their adaptation to our recurrent multi-agent setting are provided in Appendix A.

Batch-Size Scaling Setup. To examine the effect of policy–value learning formulation on large-batch training, we evaluate CEC and DCEC across multiple batch sizes in Overcooked. Here, batch size refers to the number of training samples in each PPO minibatch. At each training iteration, tasks

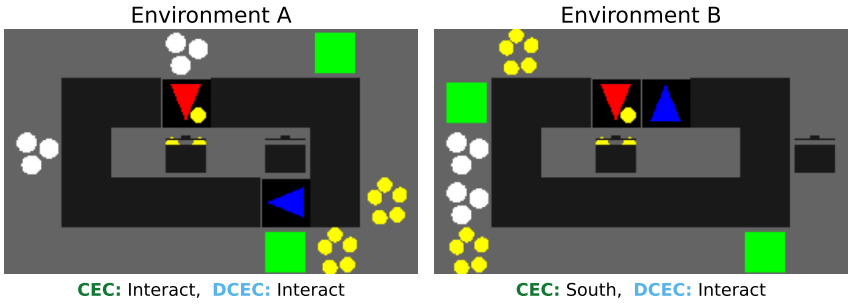


Figure 2: **Example of policy inconsistency.** In both environments, the agent is positioned in front of the pot and should choose *Interact* to continue the same subtask. CEC changes its action across the two environments, whereas DCEC maintains the same decision.

are independently sampled from the training distribution $\mathcal{M}$, and trajectories are collected in parallel. In practice, we increase the batch size by increasing the number of parallel environments N_{env}, while keeping the rollout length and the number of minibatches fixed. $N_{\text{env}} \in \{32, 64, 128, 256\}$ corresponds to batch sizes $\{8\text{K}, 16\text{K}, 32\text{K}, 65\text{K}\}$. All configurations use the same total number of environment steps.

5 Analyzing DCEC

In this section, we analyze the effects of DCEC on policy representations and cross-environment behavior compared to shared CEC. We further examine how DCEC and CEC differ in the value-learning signal as the training batch size increases.

5.1 Cross-Environment Policy Consistency

In a shared policy-value architecture, the policy and value function use the same representation, so environment-specific features learned for value estimation also influence the policy representation. As a result, the policy becomes sensitive to environment differences that are not necessary for selecting the current action. Figure 2 illustrates this issue with two states from different environments. In Environment A, the agent is holding an onion in front of the pot and selects ***Interact***. In Environment B, the same action is required, but CEC selects ***South*** instead. In contrast, DCEC selects ***Interact*** in both environments.

To examine how much environment-specific information is present in the learned representations, we conduct an environment probe analysis. In our cross-environment training setup, each procedural environment is generated from one of the five standard Overcooked layouts while task-object placements, initial agent positions, and map orientations are randomized. We refer to the underlying

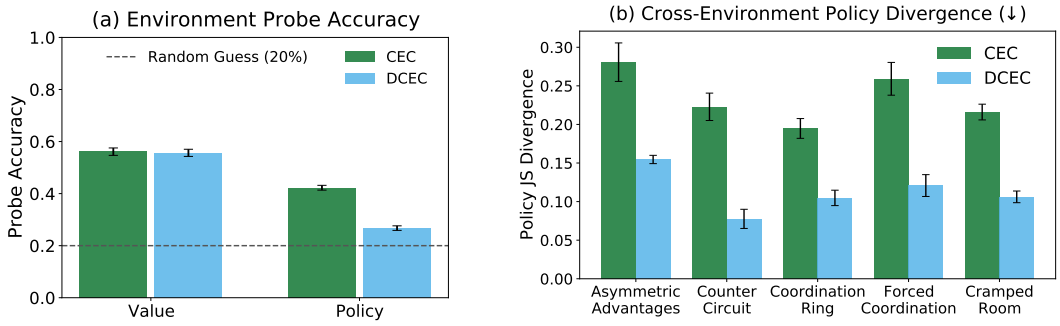


Figure 3: **Policy representation and behavior analysis.** (a) Base-layout prediction accuracy from penultimate value and policy representations. (b) Jensen–Shannon divergence between action distributions for cross-environment state pairs. All analyses use models trained with a 16K batch size. Error bars show the standard error of the mean across six seeds.

standard layout as the **base layout**. For each trained model, we extract the penultimate policy and value representations from rollout trajectories and train a linear probe to predict the base layout, evaluating it on held-out rollout episodes. Probe accuracy measures how well the base layout can be predicted from each representation. With five base layouts, random guessing accuracy is 20%. Further implementation details are provided in Appendix E.1.

Figure 3(a) shows that the value representations of CEC and DCEC contain similar amounts of base-layout information. In contrast, the CEC policy representation contains substantially more base-layout information than the DCEC policy representation. To examine whether this difference in policy representations is reflected in action selection across environments, we construct state pairs from different procedural environments in which the agent faces the same task situation and should take the same action. We measure policy consistency using the Jensen-Shannon divergence between the action distributions for each state pair, where lower divergence indicates more consistent decisions across environments. Details of subtask definitions are provided in Appendix E.2.

Figure 3(b) shows that DCEC consistently achieves lower divergence across all five base layouts, indicating more consistent action selection across environment variations. These results suggest that policy–value decoupling is associated with less environment-specific information in the policy representation and more consistent cross-environment behavior. These patterns are consistent with our hypothesis that shared policy–value learning can contribute to cross-environment overfitting.

5.2 Gradient Signal Across Batch Sizes

Having examined the analysis of policy representations and behavior, we next investigate whether DCEC benefits more from larger training batches by analyzing the parameter-wise gradient signal-to-noise ratio (GSNR) of the value-loss gradients. A higher GSNR indicates more consistent gradients across training samples for a given parameter and is related to better generalization (Liu et al., 2020). Here we use it as a diagnostic of value-learning stability across diverse training environments.

The GSNR of parameter ϕ_j is defined as $r(\phi_j) = (\mathbb{E}[g(\phi_j)])^2 / \mathrm{Var}[g(\phi_j)]$, where $g(\phi_j)$ denotes the per-sample value-loss gradient with respect to ϕ_j. The expectation and variance are estimated from per-sample gradients in the first training minibatch. For this analysis, we compute GSNR throughout the first 300 million environment steps of training and average the measurements over training and across parameters. Figure 4 compares the value-gradient GSNR of the CEC and DCEC across training batch sizes. The two architectures show similar GSNR at smaller batch sizes. In contrast, DCEC shows substantially higher GSNR at batch sizes of 32K and 65K. The gap widens as the batch size increases, indicating that DCEC provides a more consistent value-learning signal under larger-batch cross-environment training.

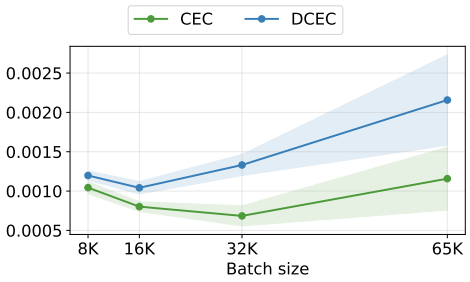


Figure 4: **Value-gradient GSNR across training batch sizes.** Higher GSNR reflects a stronger gradient signal relative to gradient noise. Shaded regions show the standard error of the mean across seeds.

6 Experiments

6.1 Experiment Setting

Environment. We evaluate DCEC in two cooperative environments: Dual Destination and Overcooked-AI. Dual Destination is a fully observable gridworld environment in which two agents coordinate to occupy the two designated goal cells. We extend the Dual Destination environment with two layout variants: *Empty* and *Wall*, and procedurally randomize task configurations during training. Overcooked-AI is a fully observable cooperative cooking environment with five standard layouts: *Cramped Room*, *Asymmetric Advantages*, *Coordination Ring*, *Forced Coordination*, and *Counter Circuit*. Following the cross-environment training setup (Jha et al., 2025), we train on procedurally generated variants of these layouts. Detailed environment specifications are provided in Appendix B.

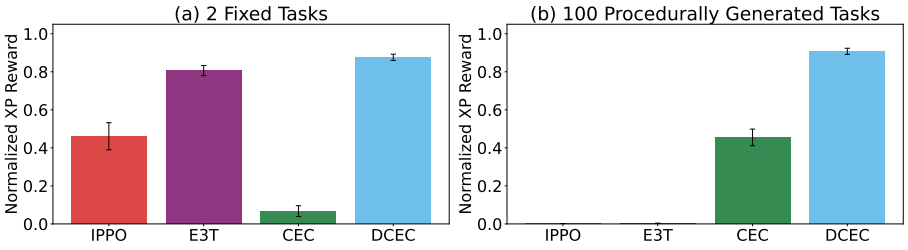


Figure 5: **Cross-play performance on Dual Destination.** Cross-play evaluation on the **(a)** 2 fixed tasks and **(b)** procedurally generated tasks. Rewards are normalized using the worst return(-100) and best return(200). Error bars show the standard error across six training seeds, averaged over layouts.

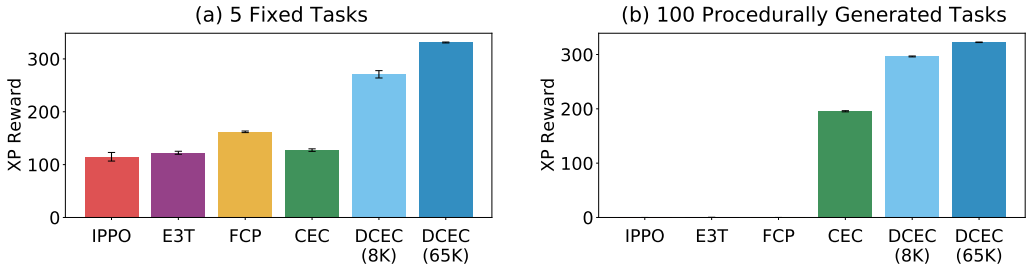


Figure 6: **Cross-play performance on Overcooked.** Cross-play evaluation on **(a)** 5 fixed tasks and **(b)** 100 held-out procedurally generated tasks. Error bars show the standard error across six training seeds, averaged over layouts.

Baselines. We compare DCEC against single-task and cross-environment baselines. The single-task baselines include IPPO (De Witt et al., 2020), FCP (Strouse et al., 2021), and Efficient End-to-End Training (E3T) (Yan et al., 2023). Each single-task baseline is trained separately on each fixed evaluation task. The cross-environment baselines include CEC (Jha et al., 2025). All baselines are trained with six random seeds. For batch-scaling experiments, CEC and DCEC use the same total number of environment steps. In Overcooked, CEC is trained with a batch size of 16K under the default cross-environment training setting. Detailed hyperparameter settings for each environment are provided in Appendix C.

Zero-shot cross-play evaluation protocol. We evaluate both fixed and procedurally generated tasks in each environment. In Dual Destination, the fixed-task evaluation uses the two layouts introduced above, while the procedural evaluation uses 100 novel generated tasks with unseen configurations. In Overcooked-AI, the fixed-task evaluation uses the five standard layouts, and the procedural evaluation uses 100 novel generated tasks that are also unseen during training. For cross-environment methods, the fixed tasks are held out from training, whereas the single-task baselines are trained separately on each corresponding fixed task.

The fixed-task evaluation measures coordination with unseen partners. The procedural evaluation tests generalization to unseen partners in unseen environments. In Dual Destination, we evaluate cross-play only with independently trained agents from different random seeds. In Overcooked-AI, we further evaluate coordination with agents trained using different algorithms and with human participants. Cross-seed evaluation measures coordination with independently learned partner policies, while cross-algorithm and human-participant evaluations assess coordination with more diverse partner behaviors.

6.2 Performance on Dual Destination Problem

Figure 5 **(a)** presents performance on the two fixed tasks. CEC is trained across procedurally generated tasks from both layout variants, while the two fixed tasks are held out for evaluation. Its cross-play performance on these fixed tasks is lower than that of the single-task baselines. This result contrasts with the original Dual Destination setting(Jha et al., 2025), where only the agents' initial positions and goal locations vary within an empty layout, and CEC performs well. This performance gap suggests that cross-environment training becomes more challenging as the structural diversity

of training environments increases. DCEC improves cross-play performance over CEC on the fixed Tasks, showing that DCEC is beneficial under structurally diverse cross-environment training. Figure 5 **(b)** evaluates generalization on 100 held-out procedurally generated tasks. The single-task baselines perform poorly, whereas CEC generalizes substantially better through cross-environment training. DCEC further improves upon CEC and achieves the highest overall performance on unseen task configurations.

6.3 Performance on Overcooked

Figure 6**(a)** compares cross-play performance on the five standard tasks. DCEC (8K) outperforms the baselines. Increasing the training batch size from 8K to 65K further improves cross-play performance, with DCEC(65K) achieving the highest average reward across the five fixed tasks. Figure 6**(b)** evaluates generalization on 100 procedurally generated layouts. The single-task methods IPPO, E3T, and FCP achieve near-zero rewards, whereas CEC retains substantial performance, confirming the importance of cross-environment training for generalization to novel environments. DCEC(8K) further improves on CEC, while increasing the batch size to 65K leads to even higher cross-play performance. These results show that DCEC improves cross-environment generalization and benefits more from larger-batch training.

To examine cross-algorithm coordination, we construct an empirical meta-game in which each strategy corresponds to a trained model, and the payoff is its cross-play reward. Following prior empirical game-theoretic analyses (Tuyls et al., 2018; Serrino et al., 2019), Figure 7 visualizes the replicator dynamics of three-strategy meta-games. We conduct this analysis on both the five fixed tasks and the 100 procedurally generated tasks. Each point in a triangle represents a population composition over three algorithms, and the arrows indicate how this composition changes under the replicator dynamics. Across the six meta-games, the dynamics generally point toward the DCEC(65K) vertex, indicating strong cross-play performance against agents trained with different algorithms. This pattern holds across comparisons with both single-task and cross-environment baselines. The full cross-algorithm performance matrices used to construct these meta-games are provided in Appendix D.2.

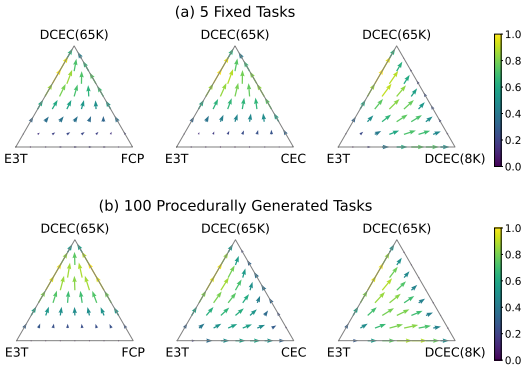


Figure 7: **Empirical game-theoretic evaluations of cross-algorithm.** Replicator dynamics on **(a)** 5 fixed tasks and **(b)** 100 procedurally generated tasks. Each triangle represents a three-strategy meta-game between different algorithms, and arrows indicate the direction of the replicator dynamics.

6.4 Large batch Cross-Environment Training

Figure 8 compares the shared and decoupled architectures across increasing training batch sizes. For CEC, performance does not improve with larger batches, remaining similar from 8K to 32K and decreasing at 65K. In contrast, DCEC outperforms CEC at 8K and continues to improve as the batch size increases, reaching its highest performance at 32K and 65K. These gains do not result from additional training experience. These results indicate that larger training batches provide limited benefits for CEC, whereas DCEC benefits more consistently from increasing the batch size. The improvement at larger batch sizes aligns with the stronger and more consistent value-learning gradients observed in Figure 4.

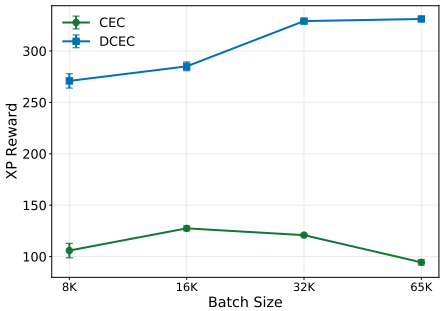


Figure 8: **Cross-play performance in 5 fixed tasks across batch sizes.** Error bars show the standard error across six training seeds, averaged over layouts.

6.5 HUMAN-AI COORDINATION

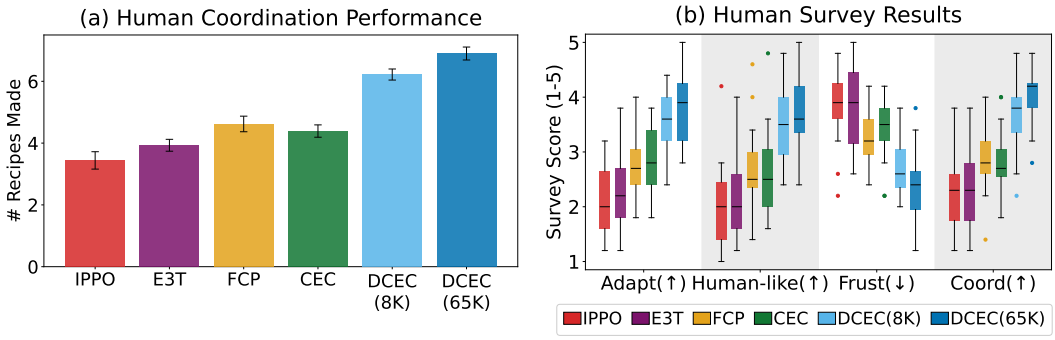


Figure 9: **Human–AI coordination evaluation.** (a) Human coordination performance is averaged across the five layouts for each participant. Error bars show the standard error of the mean across 20 participants. (b) Human survey results. Scores are averaged across five layouts per participant, with DCEC (65K) receiving significantly different ratings from all baselines after Holm correction ($p < 0.001$).

To evaluate coordination with human partners, we conduct a user study with 20 participants (10 female and 10 male, aged 20-34). Each participant interacts with all six agent variants across five Overcooked layouts, with the agent–layout combinations presented in a randomized order. Each participant completes 6 agents $\times$ 5 layouts $=$ 30 games.

Figure 9 shows the human–AI coordination results. DCEC achieves higher human coordination performance than the baseline agents, with further improvement as the batch size increases from 8K to 65K. The human survey results show a similar trend, with DCEC receiving higher ratings for adaptiveness, human-likeness, and coordination, and lower ratings for frustration. These results suggest that DCEC improves coordination with human partners in both task performance and participants' subjective evaluations. Appendix F provides further details on participant recruitment, the experimental procedure, and the survey questions.

7 CONCLUSION

In this work, we investigated whether overfitting limits cross-environment training for zero-shot coordination. Motivated by prior work showing that shared policy-value representations hinder generalization in single-agent RL (Raileanu & Fergus, 2021), we incorporate policy-value decoupling into the CEC framework and introduce DCEC. As DCEC follows the full IDAAC formulation, our results reflect the combined effects of network decoupling and the auxiliary objectives. Our analysis further shows that DCEC encodes less environment-specific information in its policy representation and makes more consistent decisions across cross-environment state pairs, supporting the hypothesis that shared policy–value learning contributes to cross-environment overfitting. Our experiments show that DCEC generalizes better to unseen partners and environments than CEC. This improvement is observed in cross-seed evaluation on Dual Destination and extends to cross-seed, cross-algorithm, and human–AI coordination in Overcooked-AI, where DCEC also benefits substantially from larger training batches.

The main limitation of our work is that we only consider environment diversity in cross-environment training. We do not consider settings that jointly incorporate environment and partner diversity. Future work should therefore investigate architectures that can effectively learn from both sources of diversity. Jointly learning across diverse environments and partners may require a more general framework that can effectively handle both sources of diversity while maintaining generalizable coordination. Developing such a framework that jointly scales environment and partner diversity is an important direction toward more general and scalable zero-shot coordination.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript drafting and language editing, literature discovery, analysis planning, interpretation of experimental results and figures, and parts of the analysis and figure-generation code. AI tools were not used to generate synthetic datasets or prove mathematical claims. The authors reviewed and validated all AI-assisted outputs and take full responsibility for the final manuscript.

ETHICS STATEMENT

This work includes a human–AI coordination study with 20 participants. Our Institutional Review Board(IRB) reviewed and approved the study protocol, and all participants provided informed consent before participating. Participants were compensated in accordance with institutional guidelines. Agent identities were not disclosed to participants during the study to reduce potential bias. We report only aggregated experimental and survey results in the paper.

REPRODUCIBILITY STATEMENT

We provide implementation details of the decoupled policy optimization in Appendix A, detailed descriptions of the environments and procedural task generation in Appendix B, and network architectures and training hyperparameters in Appendix C. Additional cross-play results are reported in Appendix D, and the human-study procedure is described in Appendix F. Agent-training experiments use six random seeds, and the human study includes 20 participants.

REFERENCES

Wilka Carvalho, Vikram Goddla, Ishaan Sinha, Hoon Shin, and Kunal Jha. Nicewebrl: A python library for human subject experiments with reinforcement learning environments. *arXiv preprint arXiv:2508.15693*, 2025.

Rujikorn Charakorn, Poramate Manoonpong, and Nat Dilokthanakul. Generating diverse cooperative agents by learning incompatible policies. In *The Eleventh International Conference on Learning Representations*, 2023.

Paresh Chaudhary, Yancheng Liang, Daphne Chen, Simon S Du, and Natasha Jaques. Improving human-ai coordination through online adversarial training and generative models. *arXiv preprint arXiv:2504.15457*, 2025.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International conference on machine learning*, pp. 1282–1289. PMLR, 2019.

Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. *arXiv preprint arXiv:2110.02034*, 2021.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in neural information processing systems*, 32, 2019.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020.

Kunal Jha, Wilka Carvalho, Yancheng Liang, Simon S Du, Max Kleiman-Weiner, and Natasha Jaques. Cross-environment cooperation enables zero-shot multi-agent coordination. *arXiv preprint arXiv:2504.12714*, 2025.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pp. 4940–4950. PMLR, 2021.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020.

Benjamin Li, Shuyang Shi, Lucia Romero, Huao Li, Yaqi Xie, Woojun Kim, Stefanos Nikolaidis, Charles Lewis, Katia Sycara, and Simon Stepputtis. Adaptively coordinating with novel partners via learned latent strategies. *Advances in Neural Information Processing Systems*, 38:1268–1295, 2026.

Yancheng Liang, Daphne Chen, Abhishek Gupta, Simon S Du, and Natasha Jaques. Learning to cooperate with humans using generative agents. *Advances in Neural Information Processing Systems*, 37:60061–60087, 2024.

Rensis Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):1–55, 1932.

Jinlong Liu, Guoqing Jiang, Yunzhi Bai, Ting Chen, and Huayan Wang. Understanding why neural networks generalize well through gsnr of parameters. *arXiv preprint arXiv:2001.07384*, 2020.

Yi Loo, Akshunn Trivedi, and Malika Meghjani. Efficient generation of diverse cooperative agents with world models. *arXiv preprint arXiv:2506.07450*, 2025.

Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. Trajectory diversity for zero-shot coordination. In *International conference on machine learning*, pp. 7204–7213. PMLR, 2021.

Kevin R McKee, Joel Z Leibo, Charlie Beattie, and Richard Everett. Quantifying the effects of environment and population diversity in multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 36(1):21, 2022.

Seungyong Moon, JunYeong Lee, and Hyun Oh Song. Rethinking value function learning for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 34846–34858, 2022.

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In *International Conference on Machine Learning*, pp. 8787–8798. PMLR, 2021.

Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862*, 2020.

Constantin Ruhdorfer, Matteo Bortoletto, Anna Penzkofer, and Andreas Bulling. The overcooked generalisation challenge. *arXiv preprint arXiv:2406.17949*, 2024.

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.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Gardar Ingvarsson, Timon Willi, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert Tjarko Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktaschel, Chris Lu, and Jakob Nicolaus Foerster. Jaxmarl: Multi-agent rl environments in jax. *arXiv preprint arXiv:2311.10090*, 2023.

Bidipta Sarkar, Andy Shih, and Dorsa Sadigh. Diverse conventions for human-ai collaboration. *Advances in neural information processing systems*, 36:23115–23139, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jack Serrino, Max Kleiman-Weiner, David C Parkes, and Josh Tenenbaum. Finding friend and foe in multi-agent games. *Advances in Neural Information Processing Systems*, 32, 2019.

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.

Karl Tuyls, Julien Perolat, Marc Lanctot, Joel Z Leibo, and Thore Graepel. A generalised method for empirical game theoretic analysis. *arXiv preprint arXiv:1803.06376*, 2018.

Caroline Wang, Arrasy Rahman, Jiaxun Cui, Yoonchang Sung, and Peter Stone. Rotate: Regret-driven open-ended training for ad hoc teamwork. *arXiv preprint arXiv:2505.23686*, 2025.

Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. *Advances in Neural Information Processing Systems*, 33: 7968–7978, 2020.

Xue Yan, Jiaxian Guo, Xingzhou Lou, Jun Wang, Haifeng Zhang, and Yali Du. An efficient end-to-end training approach for zero-shot human-ai coordination. *Advances in neural information processing systems*, 36:2636–2658, 2023.

Won-Sang You, Tae-Gwan Ha, Seo-Young Lee, and Kyung-Joong Kim. Automatic curriculum design for zero-shot human-ai coordination. *IEEE Access*, 2025.

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. In *The Eleventh International Conference on Learning Representations*, 2023.

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.

APPENDIX

A Decoupled Policy Optimization Details with IDAAC **14**

B Environment Details **15**

B.1 Cross-Environment Dual Destination 15

B.2 Overcooked 15

C Experimental Details **16**

C.1 Network Architecture 16

C.2 Hyperparameters 17

D Additional Performance Results **18**

D.1 Cross Play across different seeds 18

D.2 Cross-Play with Different Algorithms 18

E Cross-Environment Analysis Details **19**

E.1 Environment Probe Analysis 19

E.2 Cross-Environment Policy Consistency Analysis 20

F User Study Details **22**

F.1 User Study Design 22

F.2 Human–AI Behavior Analysis 24

A Decoupled Policy Optimization Details with IDAAC

As described in Section 4, we adopt the IDAAC-based policy optimization scheme (Raileanu & Fergus, 2021). The policy network is trained using the PPO (Schulman et al., 2017) objective together with advantage and instance-invariance objectives. For the instance-invariance objective, we pair policy representations at timesteps t and $t+1$ from the same agent within an episode.

The policy network is trained to maximize the following objective

$$\mathcal{J}_\pi(\theta) = \mathcal{J}_{\text{PPO}}(\theta) + \alpha_s \mathcal{S}_\pi(\theta) - \alpha_a \mathcal{L}_A(\theta) - \alpha_i \mathcal{L}_E(\theta), \tag{5}$$

where $\mathcal{J}_{\text{PPO}}$ is the PPO policy objective, $\mathcal{S}_\pi$ is the entropy bonus that encourages exploration, $\mathcal{L}_A$ is the advantage loss, and $\mathcal{L}_E$ is the instance-invariance loss. The coefficients α_s, α_a, and α_i determine the contribution of each term to the policy objective.

Advantage loss. For each agent i, we compute the generalized advantage estimate

$$\hat{A}_t^i = \sum_{k=t}^{H-1} (\gamma\lambda)^{k-t} \delta_k^i, \qquad \delta_t^i = r_t + \gamma V_\phi(o_{t+1}^i) - V_\phi(o_t^i), \tag{6}$$

where r_t is the shared reward and V_ϕ is the independently optimized value network. The policy network contains an additional advantage-prediction head A_θ, which is trained using

$$\mathcal{L}_A(\theta) = \mathbb{E}_{t,i}\left[\left(A_\theta(o_t^i, a_t^i) - \hat{A}_t^i\right)^2\right]. \tag{7}$$

The advantage loss provides an additional learning signal to the policy network.

Instance-invariant representation. To encourage the policy representation to be invariant to temporal-order information, an auxiliary discriminator is used. In our recurrent multi-agent setting, for each agent i, we form a pair using the policy representations $h_t^{\pi,i}$ and $h_{t+1}^{\pi,i}$ at timesteps t and $t+1$ within an episode. We randomly preserve or reverse their order and define as

$$q_t^i = \begin{cases} [h_t^{\pi,i},\, h_{t+1}^{\pi,i}], & y_t^i = 0, \\ [h_{t+1}^{\pi,i},\, h_t^{\pi,i}], & y_t^i = 1, \end{cases} \tag{8}$$

where $y_t^i \in \{0, 1\}$ indicates whether the pair is reversed. We construct these pairs only within the same episode, excluding transitions that cross episode boundaries and the final timestep of each rollout.

The discriminator D_ψ predicts whether the temporal order of the pair has been reversed. It is optimized using

$$\mathcal{L}_D(\psi) = \mathbb{E}_{t,i}\left[-y_t^i \log D_\psi(q_t^i) - (1 - y_t^i) \log\left(1 - D_\psi(q_t^i)\right)\right]. \tag{9}$$

The policy network is encouraged to learn representations so that the discriminator cannot identify whether the pair is in the original or reversed order.

$$\mathcal{L}_E(\theta) = \mathbb{E}_{t,i}\left[-\frac{1}{2}\log D_\psi(q_t^i) - \frac{1}{2}\log\left(1 - D_\psi(q_t^i)\right)\right]. \tag{10}$$

When minimizing $\mathcal{L}_D$, only the discriminator parameters are updated, while the policy encoder remains fixed. Similarly, when minimizing $\mathcal{L}_E$, only the policy encoder parameters are updated, while the discriminator remains fixed. Minimizing $\mathcal{L}_E$ encourages $D_\psi(q_t^i)$ toward 0.5. This makes the ordering difficult to infer from the policy representation. We adapt the temporal-order objective of IDAAC to our recurrent multi-agent setting

Implementation details. We set the advantage auxiliary coefficient to $\alpha_a = 0.25$ and the instance-invariance coefficient to $\alpha_i = 0.001$. We optimize the policy, value, and discriminator parameters using separate Adam optimizers, each with an initial learning rate of 3×10^{-4} that is linearly annealed over training.

B Environment Details

B.1 Cross-Environment Dual Destination

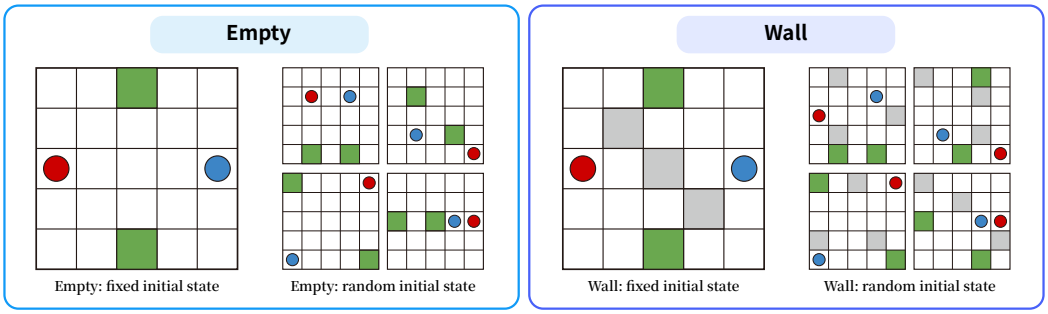


Figure 10: Overview of the Dual Destination Problem layouts.

Dual Destination is a simple, fully observable cooperative gridworld in which two agents receive a shared reward for occupying two designated goal cells. Both agents receive a $+3$ reward for reaching the designated goal cells. At each timestep, each agent can move up, down, left, or right, or stay in place, and receives a step -1 penalty.

Fixed tasks. We use two Dual Destination layouts: *Empty* and *Wall*, with obstacles arranged diagonally across the grid. The two fixed configurations are excluded from cross-environment training and used as held-out evaluation tasks, as illustrated in Figure 10.

Procedurally generated tasks. For cross-environment training, we uniformly sample one of the two layout variants at the beginning of each episode. Given the selected layout, the agents' initial positions, wall locations, and the two goal locations are randomly sampled to generate a new task configuration. For evaluation, we use 100 procedurally generated task configurations that are held out from training.

B.2 Overcooked

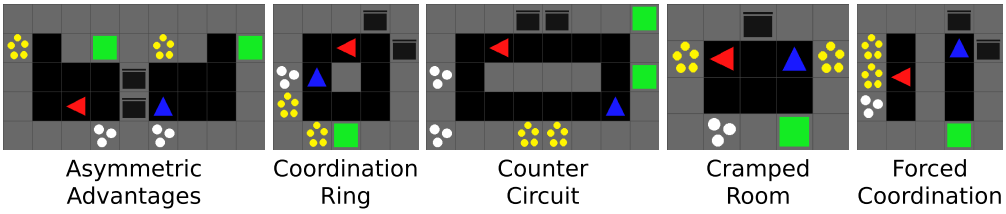


Figure 11: Overview of the Overcooked layouts used in our experiments.

Overcooked-AI is a fully observable two-agent cooperative environment in which agents collaborate in a grid-world kitchen to prepare and deliver soups. Each agent has a discrete action space with six actions: move up, down, left, or right, interact, and stay. The sparse task reward is $+20$ for each successfully delivered soup. During training, we additionally use shaped rewards for subtask actions. Placing an onion in a pot and picking up a plate each provide $+3$, while picking up a cooked soup provides $+5$. Our implementation follows the JaxMARL[1] (Rutherford et al., 2023) version of Overcooked-AI.

Fixed tasks. We use five standard Overcooked layouts: Cramped Room, Asymmetric Advantages, Coordination Ring, Forced Coordination, and Counter Circuit. Each layout specifies a fixed

[1] https://github.com/bold-lab-ai/JaxMARL

wall structure and the original locations of objects, including onion piles, plate piles, pots, and serving locations, as well as the initial agent positions. These five standard layouts are used as fixed evaluation tasks, as illustrated in Figure 11.

Procedurally generated tasks. Following CEC (Jha et al., 2025), training tasks are procedurally generated from the wall structures of the five standard Overcooked layouts. Onion piles, plate piles, pots, serving locations, and agent positions are resampled while ensuring task solvability. The generated grid is randomly rotated 90° clockwise and then padded with walls to fit a 9×9 grid. To keep the five standard configurations held out from training, each generated task is checked against them. If the onion pile, plate pile, pot, and serving locations exactly match any of the standard configurations, the task is discarded and resampled.

C Experimental Details

C.1 Network Architecture

Base network architecture. IPPO, FCP, CEC, and DCEC use the network dimensions shown in Table 1. The network consists of an observation encoder, a recurrent core, and actor and critic heads. IPPO, FCP, and CEC share the observation encoder and recurrent representation between the actor and critic.

Component	Layer	Details
Observation encoder	Conv1	2×2 kernel, 64 filters, ReLU
	Conv2	2×2 kernel, 32 filters, ReLU
	FC1	512 units, ReLU
	FC2	512 units, ReLU
Recurrent core	LSTM	256 hidden units
Actor head	FC1	256 units, ReLU
	FC2	192 units, ReLU
	FC3	128 units, ReLU
	FC4	64 units, ReLU (Overcooked only)
	Output	$\lvert\mathcal{A}\rvert$ action logits
Critic head	FC1	512 units, ReLU
	FC2	256 units, ReLU
	FC3	192 units, ReLU (Overcooked only)
	FC4	128 units, ReLU (Overcooked only)
	Output	1 value prediction

Table 1: Base network architecture used in our experiments. All layers use orthogonal initialization and zero bias initialization.

Decoupled architecture. DCEC uses separate policy and value networks with the same layer dimensions as the base architecture. The policy network uses the actor head together with an advantage-prediction head and a temporal-order discriminator. The advantage head predicts one value for each discrete action, while the discriminator predicts the ordering of policy representations to encourage instance-invariant representations. Both auxiliary modules are used only during training, as described in Appendix A.

E3T architecture. E3T uses the same base encoder, recurrent core, and actor-critic head dimensions, with an additional partner-action predictor following Yan et al. (2023). This predictor estimates the partner's action, and the actor conditions its policy on this prediction to adapt to the partner's behavior. We set the randomness parameter of the partner policy to 0.55.

C.2 HYPERPARAMETERS

Table 2: Common PPO hyperparameter settings for the Dual Destination and Overcooked experiments.

Hyperparameter	**Dual Destination Problem**	**Overcooked v1**
Optimizer	Adam ($\epsilon = 10^{-5}$)	Adam ($\epsilon = 10^{-5}$)
Learning rate (actor / critic)	3×10^{-4}	3×10^{-4}
Discount factor γ	0.99	0.99
GAE λ	0.95	0.95
PPO clip ϵ	0.2	0.2
Entropy coef.	0.005	0.005
Value function coef.	1.0	1.0
Number of Minibatches	2	2
Steps per rollout	100	256
Update epochs	32	4
Max grad norm	0.5	0.5
Anneal learning rate	True	True
Total time steps	1e8	3e9

The batch size is defined as the number of training samples in each PPO minibatch:

$$B_{\text{mini}} = \frac{N_{\text{agents}} \times N_{\text{env}} \times T_{\text{rollout}}}{N_{\text{minibatches}}}. \tag{11}$$

Dual Destination Problem. For Dual Destination, we use a fixed training configuration across all experiments with $N_{\text{env}} = 256$, $T_{\text{rollout}} = 100$, and $N_{\text{minibatches}} = 2$, resulting in a batch size of $25{,}600$.

Overcooked v1. For Overcooked, we study the effect of increasing the training batch size by varying the number of parallel environments N_{env}, while keeping the rollout length and the number of minibatches fixed. Specifically, $N_{\text{env}} \in \{32, 64, 128, 256\}$ corresponds to batch sizes of $\{8{,}192,\ 16{,}384,\ 32{,}768,\ 65{,}536\}$, respectively. Unless otherwise specified, CEC is trained with a batch size of 16,384.

D ADDITIONAL PERFORMANCE RESULTS

D.1 CROSS PLAY ACROSS DIFFERENT SEEDS

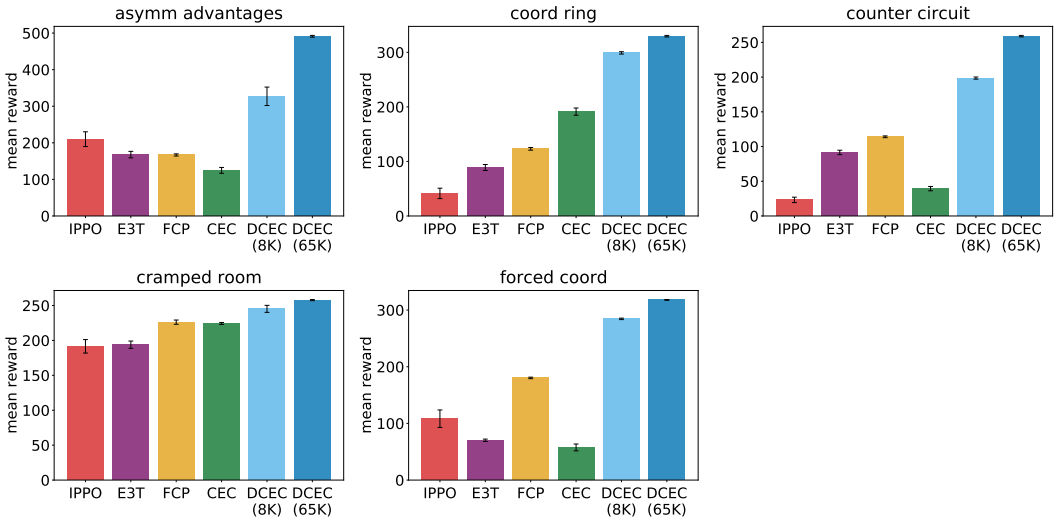


Figure 12: **Cross-seed performance across the five fixed Overcooked tasks.** Cross-play evaluation between agents trained with different random seeds across the five fixed tasks. Error bars show the standard error of the mean.

D.2 CROSS-PLAY WITH DIFFERENT ALGORITHMS

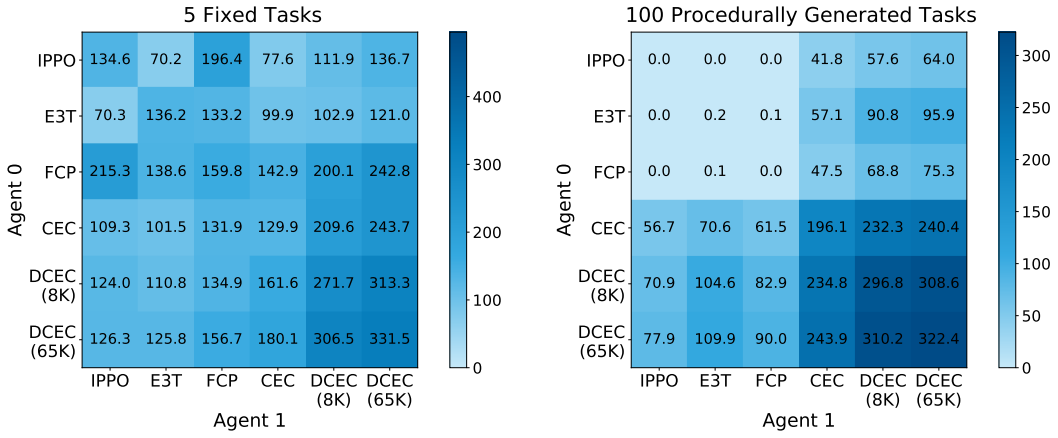


Figure 13: **Cross-algorithm evaluation across Overcooked layouts.** Cross-play performance between agents trained with different algorithms, evaluated on 5 fixed tasks (left) and 100 procedurally generated tasks (right). The cross-play matrix is asymmetric because agents start in different initial positions. Each cell shows the cross-play performance of the corresponding agent pair.

E Cross-Environment Analysis Details

E.1 Environment Probe Analysis

We use a linear probe to measure how much information about the underlying environment is present in the learned policy and value representations. We consider five base layouts: Asymmetric Advantages, Coordination Ring, Counter Circuit, Forced Coordination, and Cramped Room. For each base layout, procedural environments are generated by randomizing task-object placements, initial agent positions, and map orientations.

Representation extraction. For each trained model and random seed, we freeze the network parameters and collect rollout trajectories from procedurally generated environments. We separately extract the penultimate policy representation h^{π} and value representation h^{V}. For each base layout, we collect 20 rollout trajectories of 128 environment steps and sample 32 timesteps at regular intervals from each trajectory. Representations from both agents are included, resulting in $20 \times 32 \times 2 = 1{,}280$ vectors per base layout and 6,400 vectors in total for each representation type.

Probe training and evaluation. We train a single-layer linear classifier on representations extracted from the frozen network to predict base layouts. Before fitting the classifier, we standardize each feature using statistics computed from the probe-training set. For each base layout, we split the 20 episodes into 14 training episodes and 6 evaluation episodes. We repeat this episode-level split 10 times and average the evaluation accuracy across the splits. We then report the mean and standard error across six training seeds. With five base layouts, random-guess accuracy is 20%.

E.2 Cross-Environment Policy Consistency Analysis

Subtask Pair Construction. To evaluate cross-environment policy consistency, we construct state pairs for five Overcooked subtasks. In each pair, the ego agent is placed in the same subtask situation. It holds the same item, faces the same type of task-relevant object, and has the same relative position and orientation with respect to that object. The paired states share the same base layout but differ in object positions and the partner position. We then compare the agent's action distributions across these state pairs to measure whether it makes consistent decisions despite these environmental differences.

Table 3 summarizes the state conditions used to define the five Overcooked subtasks, and Figure 14 shows state pairs for each subtask. For example, in the plate-pickup subtask, a ready pot is included so that collecting a plate corresponds to an active cooking situation.

For each base layout, we scan 3,000 procedurally generated environments and collect up to 50 state pairs per subtask, yielding up to 250 pairs per base layout. Pairs are constructed solely from the predefined state conditions, without using model predictions. We additionally balance the ego agent's orientation across the four cardinal directions.

Table 3: **State conditions for cross-environment state pairs.** The table summarizes the state conditions used to construct state pairs for the five Overcooked subtasks in the policy-consistency analysis.

Subtask	Ego inventory	Target condition
Plate pickup	Empty	Facing a plate dispenser
Cooked soup pickup	Plate	Facing a pot with cooked soup
Serve	Completed soup	Facing a serving station
Onion pickup	Empty	Facing an onion dispenser
Third onion to pot	Onion	Facing a pot containing two onions

Policy Consistency Evaluation. When evaluating each constructed state, we initialize the LSTM recurrent state to zero. This ensures that the paired action distributions are evaluated under the same recurrent context and excludes differences arising from their preceding trajectory histories. For each state pair (s_A, s_B), let o_A and o_B denote the corresponding ego-agent observations. We obtain

$$P = \pi(\cdot \mid o_A, z_0) \quad \text{and} \quad Q = \pi(\cdot \mid o_B, z_0).$$

where z_0 denotes the zero-initialized recurrent state. We compute their Jensen–Shannon (JS) divergence as

$$D_{\mathrm{JS}}(P, Q) = \frac{1}{2} D_{\mathrm{KL}}(P \| M) + \frac{1}{2} D_{\mathrm{KL}}(Q \| M), \qquad M = \frac{1}{2}(P + Q).$$

We first average the JS divergence within each subtask–base-layout combination. For the layout-level results, we average equally across the five subtasks for each training seed and report the mean and standard error across six seeds.

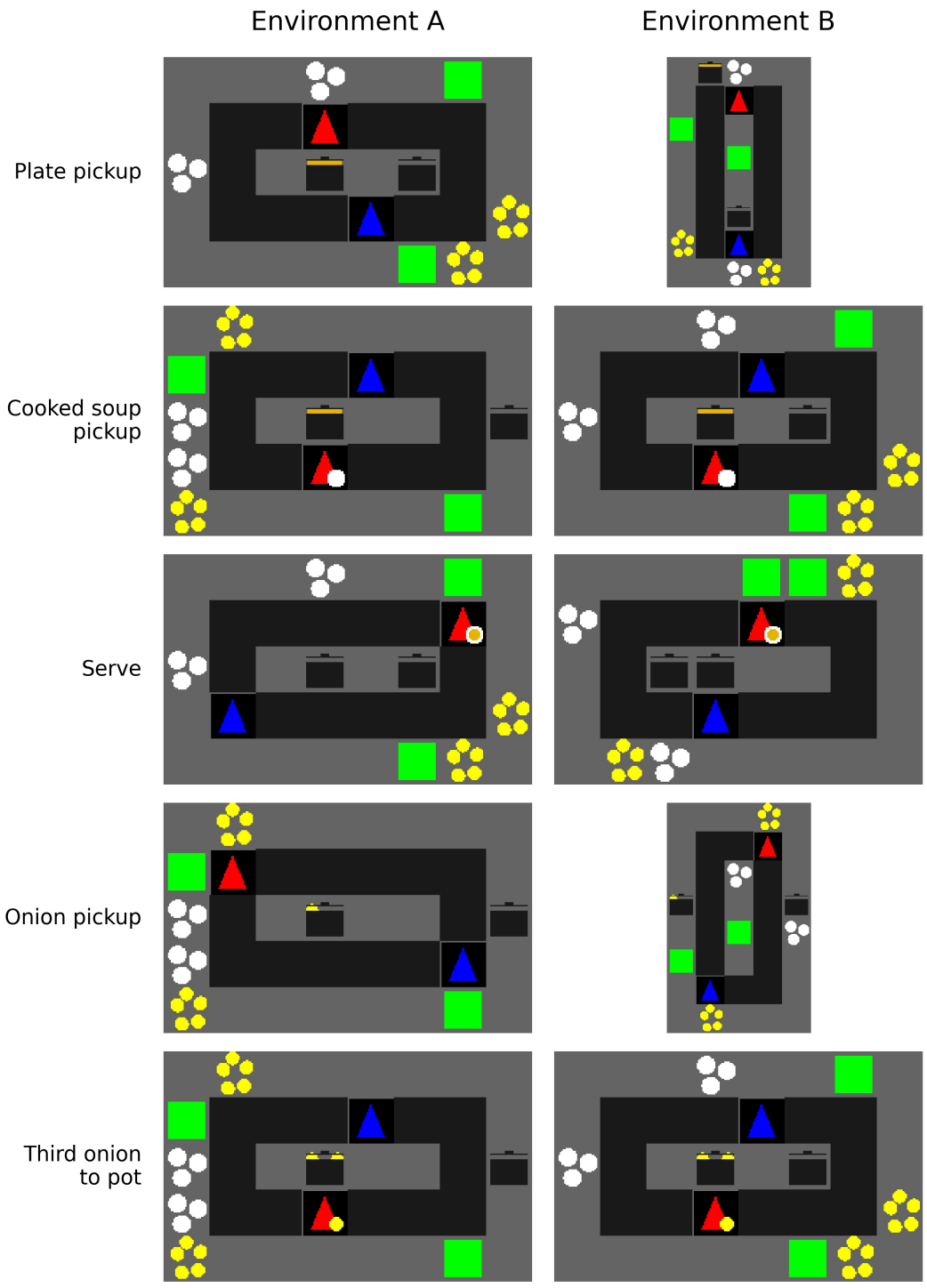


Figure 14: **Cross-environment state pairs in Counter Circuit.** Each row shows a representative pair for one of the five Overcooked subtasks, with the ego agent matched in inventory, target condition, and relative position and orientation with respect to the task-relevant object.

F User Study Details

F.1 User Study Design

We conducted a user study in which each participant interacted with all six agents, including IPPO, E3T, FCP, CEC, DCEC(8K), and DCEC(65K), across five standard Overcooked layouts: Asymmetric Advantages, Coordination Ring, Counter Circuit, Cramped Room, and Forced Coordination. Participants completed one game for each agent–layout combination, resulting in 6 agents $\times$ 5 layouts $=$ 30 episodes. For each game, we randomly assigned the participant and agent to the two starting positions. We also randomized the agent order for each participant to mitigate order effects. Each episode consisted of 200 environment steps, with a reward of 20 for each successful delivery. Participants completed a tutorial before the main experiment, and the full study took approximately one hour.

The study was conducted through a web-based interface built with NiceWebRL[2] Carvalho et al. (2025). NiceWebRL supports server-side parallelization of game environments and AI agents, enabling human experiments in JAX-based game environments while facilitating participant data collection and storage. A total of 20 participants (10 female and 10 male; aged 20–34) were recruited for the user study. All studies were conducted in person at our school. The study protocol was approved by our Institutional Ethics Review Board (IRB), and all participants provided informed consent. Participants were not informed of the agent identities during the study to reduce potential bias. Participants rated the agent after each interaction using seven 5-point Likert-scale questions (Likert, 1932). The first six questions used response options ranging from *strongly disagree* to *strongly agree*, while the final question used *very poor* to *very good*. The questions were:

1. The agent adapted to me when making decisions.
2. The agent was consistent in its actions.
3. The agent's actions were human-like.
4. The agent frequently got in my way.
5. The agent's behavior was frustrating.
6. Overall, I enjoyed playing with the agent.
7. Overall, I felt that the agent's ability to coordinate with me was:

For each survey question, we first averaged responses across the five layouts for each participant and agent, resulting in one score per participant for each agent. We then compared agent-level scores using pairwise paired t-tests across the 20 participants, with Holm correction for multiple comparisons. All pairwise comparisons remained statistically significant after Holm correction (adjusted $p < 0.001$). Figure 15 shows the corresponding response distributions.

[2] https://github.com/KempnerInstitute/nicewebrl

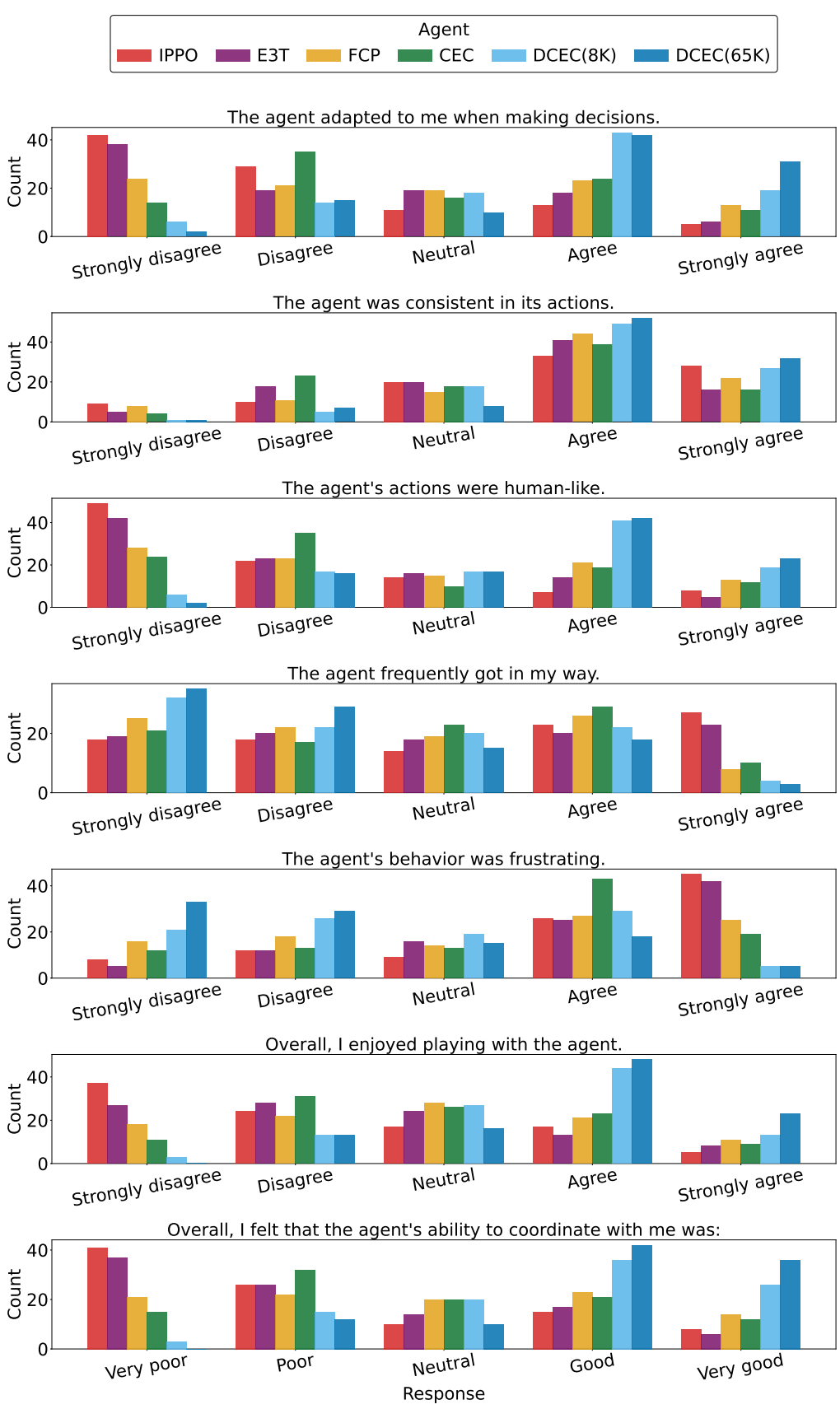


Figure 15: **Distribution of human ratings across survey questions.** Each panel shows the distribution of participant responses to one of the seven survey questions.

F.2 Human–AI Behavior Analysis

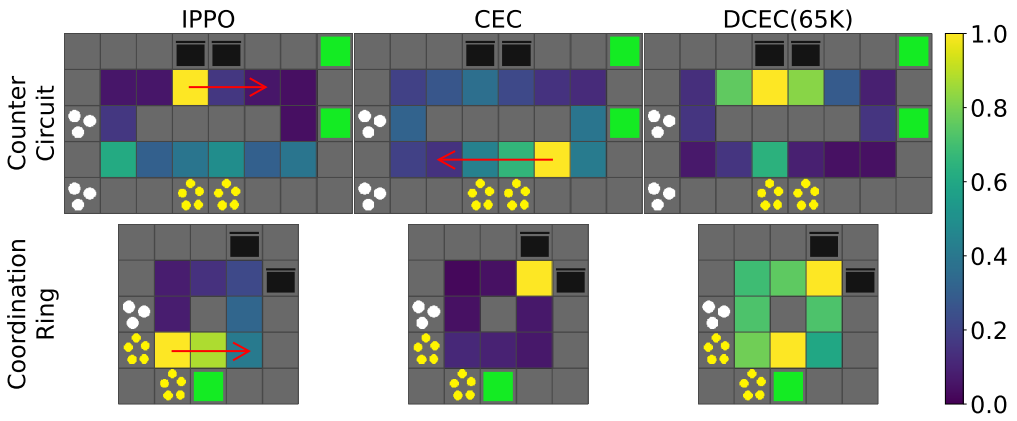


Figure 16: **Visit Frequency.** Visit frequency across grid locations for each agent and layout. Darker colors indicate lower visit frequency, while brighter colors indicate higher visit frequency.

This section analyzes the agents' movement patterns during interactions with human partners. We compare the movement patterns of IPPO, CEC, and DCEC in Counter Circuit and Coordination Ring. As shown in Figure 16, IPPO tends to move repeatedly along one dominant route in both Counter Circuit and Coordination Ring. CEC shows a similar directional pattern in Counter Circuit, while in Coordination Ring its visits are concentrated around a small region. These patterns suggest that IPPO and CEC rely on relatively fixed movement patterns during interaction with human partners. In contrast, DCEC frequently visits task-relevant locations around the pots and onion piles while using the left and right routes more evenly. This pattern is especially clear in Coordination Ring, where DCEC moves in both directions around the layout. These results suggest that DCEC exhibits more adaptive movement during human–AI interaction, with its movement distributed across multiple task-relevant routes.