Concept-based Understanding of Emergent Multi-Agent Behavior

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this work, we propose novel techniques that improve understanding of emergent multi-agent behavior via concept-based interpretability. We propose Concept Bottleneck Policies (CBPs) as a method for learning intrinsically interpretable, multi-agent policies. By constructing each agent's policy to be conditioned on its own estimates of human-understandable concepts, our method also enables novel forms of behavioral and inter-agent social analysis via concept intervention. For example, we demonstrate experimentally that our method can reliably detect: (i) the emergence of coordination, coordination failures (lazy agents), and environmental demands for coordination in cooperative environments; (ii) the emergence of skill-based strategy and role assignment in competitive environments; and (iii) emergent social behaviors such as public resource allocation and exploitation in social dilemmas. Moreover, CBPs match the performance of standard, non-concept-based policies; thereby achieving interpretability without sacrificing performance.

1 Introduction

Multi-agent learning continues to play a crucial role in the development of scalable and generally-capable AI systems. In addition to well-known successes in board games [32, 35, 36] and online games [6, 44], multi-agent learning has enabled agents to develop a range of capabilities, such as navigating social dilemmas [20], optimizing traffic flow for autonomous vehicles [43, 45], and even learning to cooperate with humans [7]. For all its success, interpreting emergent multi-agent behavior remains an open challenge. In cooperative environments, for example, reward alone is not enough to understand the nature of a learned coordination strategy, or whether agents have learned to coordinate at all. In more complex settings, such as social dilemmas, agents often learn nuanced relationships that include both low-level spatial (e.g., navigation) and high-level social (e.g., exploitation, public resource allocation) interactions that can only be fully-understood through exhaustive visualization.

For this reason, recent work has shifted focus from traditional measures of performance (i.e. reward, human analysis) to better understanding emergent behaviors [30]. Related methods perform interpretability analysis in a *post-hoc* manner, either by measuring basic behavioral statistics [23], or regressing concepts from policy network activations [27]. An alternative to post-hoc analysis is *intrinsic interpretability*, where a model's decisions incorporate human-understandable concepts directly. In supervised learning settings, for example, it has been shown that it is possible to train a neural network that is "bottlenecked" by human-understandable concepts, enabling intrinsic interpretability while maintaining high performance in classification tasks [19].

In this work, we introduce an intrinsically-interpretable, concept-based policy architecture for multi-agent learning. Our proposed architecture, the Concept Bottleneck Policy (CBP), forces an agent to make decisions by first predicting an intermediate set of human-understandable concepts, then using

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

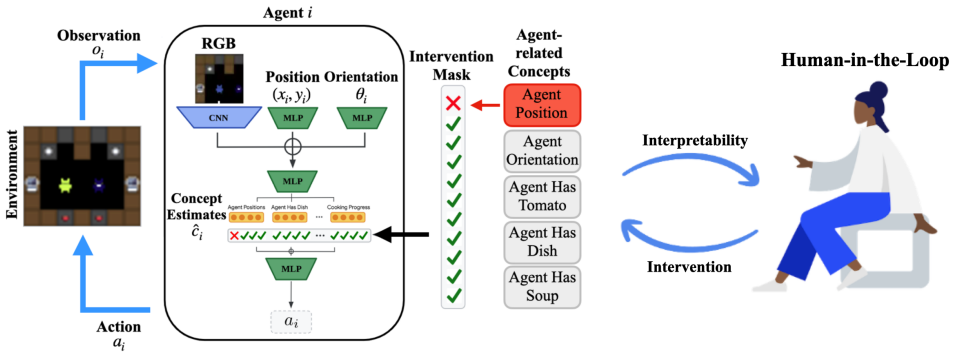


Figure 1: Concept Bottleneck Policies distill agent decisions into human-understandable concepts, yielding interpretable policies for MARL and enabling behavioral analysis via concept intervention.

those concepts to select actions. This architecture makes decision-making immediately transparent, as agents reveal the environmental and inter-agent factors influencing each of their actions.

Beyond intrinsic interpretability on a per-agent basis, our method also **unlocks a novel class of techniques for understanding emergent multi-agent behavior**. By conditioning each agent's actions on its own concepts from construction, our method supports behavioral analysis via concept intervention. Specifically, we can perform interventions over each concept estimate in an agent's CBP and examine its impact on the larger multi-agent system (with respect to rewards, actions, environment features, etc). We use this technique to uncover many key aspects of multi-agent behavior.

In cooperative environments, our method can reliably detect agents who have learned to coordinate from those that act independently, determine which environments require coordination to solve vs. those that permit independent solutions, expose inter-agent factors that drive coordination, and even diagnose common failure modes such as lazy agents. In competitive environments, our method disentangles more sophisticated strategic behavior (e.g., role assignment) as it emerges during training. In social dilemmas, we show that it is possible to identify inter-agent social dynamics (e.g., exploitation, public resource sharing) by learning a graph over intervention outcomes. Specifically, we learn a sparse graph representing the pairwise relationships of inter-agent intervention effects and show that it captures both high-level behavioral patterns (free loader agents vs. public service agents) and low-level spatial information (agents who interact). Finally, we show that CBPs perform comparably to non-concept based policies, thus achieving interpretability without sacrificing performance.

In sum, our contributions are as follows: (i) We introduce Concept Bottleneck Policies as an interpretable, concept-based architecture for MARL; (ii) We introduce a suite of novel techniques for multi-agent behavioral understanding that leverage concept intervention and graph learning; (iii) We show that CBPs can match the performance of non-concept-based network architectures.

2 Related Work

Our work lies at the intersection of interpretability and MARL. In interpretability, computer vision works have used saliency maps to help provide pixel-level explanations for model predictions [37, 38, 42, 39]. Interpretability using grounded concepts has also been explored to provide more meaningful, human-understandable explanations of model decisions [18, 14, 5, 13, 8, 48, 40]. As described earlier, concept bottleneck models [19] constrain the networks through such grounded concepts, enabling analysis of the model via intervention.

RL interpretability techniques include those that either increase the transparency of agent decision-making directly [4, 25]; or analyze agent behaviors from a post-hoc perspective [47, 2, 34, 24, 17, 30, 27, 11]. Approaches that directly increase transparency include those that combine model compression and decision trees to yield more interpretable agent policies [4], or rely upon causal decision trees with limited depth to explore causality of agent decisions [25]. However, these approaches have not yet been extended to RL settings with complex observations (e.g., images), where function approximators are typically needed. Post-hoc methods have used traditional saliency-mapping techniques to attribute decisions to image observation regions [47], highlighting states that lead to major differences in agent behaviors [2], and summarized key agent behaviors using

measures such as action uncertainty [34]. Increasingly, natural language has been used to aid human understanding of agents through instructions [1] or explanations [15].

Interpretability of MARL agents has received limited attention in prior works, with primary investigations gauging agents' internal representations by making predictions about future outcomes [24], visualizing latent clusterings of agents' neural activation vectors [17, 26], or using offline behavioral analysis to learn behavioral spaces over agents [30]. Recent work has also analyzed decision-making in two-player games such as Chess [27], and Hex [11]. In contrast to our work, these approaches focus on post-hoc interpretability, whereas ours makes agent decisions directly transparent and enables novel forms of downstream analysis. To the best of our knowledge, ours is the first work to identify coordination, lazy agents, emergent skill learning, and social behaviors via intervention.

3 Background

Markov Games A partially-observable Markov game [22] of N agents is defined by the tuple $\mathcal{M} = (\mathcal{S}, \boldsymbol{\mathcal{A}}, \mathcal{T}, \boldsymbol{\mathcal{R}}, \boldsymbol{\Omega}, \mathcal{O})$ where $\mathcal{S}$, $\boldsymbol{\Omega}$, and $\boldsymbol{\mathcal{A}}$ are the game's global state-space, joint observation-space, and joint action-space, respectively. In each state, each agent i selects an action $a_i \in \mathcal{A}_i$, yielding a joint action $\boldsymbol{a} = (a_1, ..., a_N)$ for all agents. Following action selection, the environment transitions to a new state according to the transition function $\mathcal{T} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathcal{S}$, produces new observations for each agent following the observation function $\mathcal{O} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \boldsymbol{\Omega}$, and emits a reward defined by the reward function $\boldsymbol{\mathcal{R}} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}^N$. The actions of each agent are dictated by a policy $\pi_i(a_i|o_i)$ and the collection of all individual policies $\boldsymbol{\pi} = (\pi_1, ..., \pi_N)$ is called the joint policy.

Concept Bottleneck Models CBMs are a class of intrinsically-interpretable neural network architecture for supervised learning [19]. CBMs make predictions by first estimating a set of human-understandable concepts, then producing an output based on those concepts–i.e., the network is "bottlenecked" by concepts. Formally, given a dataset $\{(x^{(j)}, y^{(j)}, c^{(j)})\}_{j=1}^{n}$ consisting of inputs $x \in \mathbb{R}^d$, outputs $y \in \mathbb{R}$, and human-understandable concepts $c \in \mathbb{R}^k$ (where k is the number of unique concepts), a concept bottleneck model learns two mappings: $f : \mathbb{R}^d \to \mathbb{R}^k$ from input-space to concept-space, and $g : \mathbb{R}^k \to \mathbb{R}$ from concept-space to output-space. The model can then make predictions $\hat{y} = f(g(x))$ as a composition of those mappings. For example, in an application of arthritis detection from MRI images, the model first predicts related concepts such as knee joint spacing and the presence of bone spurs, then uses those concepts to classify arthritis severity.

4 Concept Bottleneck Policies for MARL

Here we introduce Concept Bottleneck Policies (CBPs) as an intrinsically interpretable, concept-based policy learning method for MARL (see Figure 1). First, we extend the Markov game formalism to support concept-based learning. Then we introduce the CBP architecture and show how it fits within any policy learning scheme. Finally, we demonstrate our method's potential for interpreting emergent multi-agent behavior, introducing multiple behavioral analysis techniques that use intervention to detect a variety of multi-agent social phenomena.

Concept-based Markov Games We extend Markov games in two important ways. First, we assume that, in addition to a state space $\mathcal{S}$, there exists an interpretable concept-state space $\mathcal{C}$, where each concept-state $\boldsymbol{c} \in \mathcal{C}$ is a vector of human-understandable concepts that describe key features of the environment (e.g., agent positions, object states). This assumption holds in many RL domains like Atari [29] and Melting Pot [21]. Second, we allow agents to generate estimates $\hat{\boldsymbol{c}}$ of the concept state as part of their internal representation. In sum, at each time-step t, the environment produces both a state s_t and concept-state $\boldsymbol{c}_t$, and each agent i selects both actions $a_{i,t}$ and concept estimates $\hat{\boldsymbol{c}}_{i,t}$.

Concept Bottleneck Architecture There are many ways in which concepts can be modeled within the RL framework. Inspired by CBMs [19], we examine policy architectures in which an agent's action is conditioned entirely on its own concept estimates. To this end, we factorize an agent i's policy $\pi_i(a_i|o_i)$ into a function $f_i : \mathcal{O}_i \to \mathcal{C}_i$ mapping observations o_i to concept estimates $\hat{\boldsymbol{c}}_i$; and $\pi_i^{\text{act}} : \mathcal{C}_i \to \mathcal{A}_i$ mapping concept estimates $\hat{\boldsymbol{c}}_i$ to actions a_i. Composing f_i and π_i^{act} yields a standard policy mapping observations to actions: $\pi_i(a_i|o_i) = \pi_i^{\text{act}}(f(o_i)) = \pi^{\text{act}}(a_i|\hat{c}_i)$. Given an observation $o_{i,t}$ for some agent i and time t, the bottleneck first produces concept estimates $\hat{\boldsymbol{c}}_{i,t} = f(o_{i,t})$, then uses those estimates alone to select an action $a_{i,t} \sim \pi_i^{\text{act}}$. Conditioning actions on concepts creates an intrinsically interpretable policy, as the policy is forced to provide a human-understandable rendering

of the factors driving its decision making. Importantly, $\hat{\boldsymbol{c}}_{\boldsymbol{i}}$ can be also used to interpret multi-agent behavior. For example, if two agents i and j collide while moving in the environment, examining $\hat{\boldsymbol{c}}_i$ and $\hat{\boldsymbol{c}}_j$ may identify which of the agents incorrectly modeled the location of its teammate.

Concept Bottleneck Learning Under the CBP framing, learning a strong but interpretable policy requires both (i) learning to predict concepts accurately, and (ii) learning to select actions from those concepts effectively. To achieve the former, we introduce a concept loss: $L_C(\boldsymbol{c}, \hat{\boldsymbol{c}}) = \sum_{j=1}^{|C|} L_{C_j}(c_j, \hat{c}_j)$, where each component L_{C_j} measures the error between the j'th predicted concept and its concept label. Each L_{C_j} is a supervised loss, with a specific form dictated by the value of c_j—mean-squared error if c_j is scalar, log loss if c_j is binary, etc. In practice, each pair of concepts $\boldsymbol{c}$ and concept estimates $\hat{\boldsymbol{c}}$ are stored in an agent's replay buffer during training alongside standard (s, a, r, s') tuples.

To learn a policy from concept estimates, we attach L_C as an auxiliary loss to the reward-based loss defined by any base RL algorithm (e.g., PPO [33], etc). In general, if L_{RL} is a generic reward-based loss, we construct a joint concept bottleneck policy loss $L_{CBP} = L_{\text{RL}} + \lambda L_C$, where the concept loss coefficient λ weights the relative importance of concept prediction.

Behavioral Analysis via Concept Intervention Intervention is a powerful tool for understanding the statistical relationships that exist between random variables (e.g. in a graphical model) by conducting experiments in which the value of a random variable is fixed in order to observe that variable's effect on other variables[31]. A key feature of our method is its support of this type of intervention analysis. For an agent i, let $\text{Int}(\hat{c}_j, \bar{c}_j, i)$ be an intervention over the j'th concept estimate $\hat{c}_j$ from i's bottleneck with the replacement value $\bar{c}_j$ (in the simplest case, $\bar{c}_j = 0$). We can then observe the effect, if any, that $\bar{c}_j$ has on the agent's behavior by comparing its impact on reward:

$$\mathbb{E}\left[\sum_t r(s_t, \pi_i^{\text{act}}(\hat{c_j}))\right] - \mathbb{E}\left[\sum_t r(s_t, \pi_i^{\text{act}}(\bar{c_j}))\right] \tag{1}$$

In mutual reward settings, this serves as a proxy for the intervention's impact on team behavior.[1] Here we propose two examples of behavioral tests enabled by this approach:

Detecting Coordination: Let i and j be two agents. For i to coordinate with j, i must condition its policy on information about j (j's position, orientation, etc). If the agents are coordinating and we remove i's concept estimates pertaining to j, we should expect a decrease in reward. Conversely, if reward does not degrade, then i and j must not be explicitly coordinating (i.e., directly using signals from each other). By intervening over concepts pertaining to an agent's teammates, therefore, we can identify whether or not agents are coordinating.

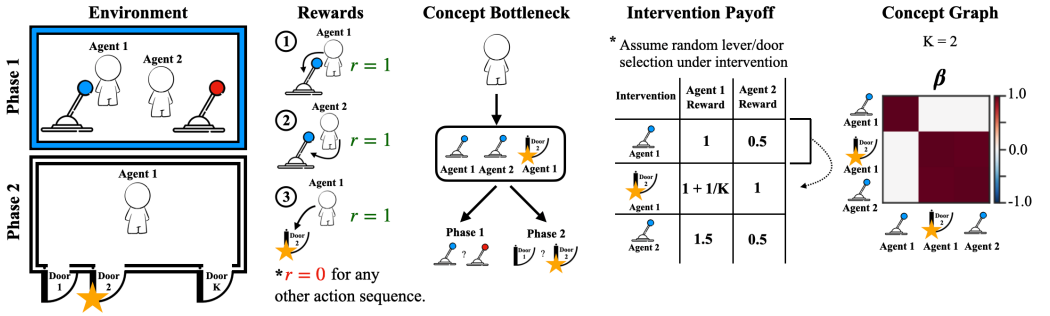
Exposing Lazy Agents: Here we define a lazy agent as one that does not contribute to increasing team reward through its own actions. For an agent to contribute, it must at least condition its policy on information about its own interactions with the environment. A lazy agent, therefore, is one that has learned a sub-optimal policy that does not encode such information, leading to unproductive behavior. It follows that we can test for the *degree of laziness* of an agent by replacing its concept estimates *about itself* and measuring the resulting degradation of team reward. If performance remains the same with the agent incapacitated, we conclude that it is a lazy agent.

Modeling Inter-Agent Social Dynamics Here we show that it is possible to deepen our intervention-based interpretability by learning a sparse graph over CBP intervention outcomes. Specifically, we take each intervention $\text{Int}(\hat{c}_j, \bar{c}_j, i)$ to be a random variable that is described by some agent-related features, such as the average reward $\boldsymbol{r}_{\hat{\boldsymbol{c}}_{\boldsymbol{j}}} = \{r^1_{\hat{c}_j}, ..., r^n_{\hat{c}_j}\}$ each agent receives under the intervention[2]. By performing $\text{Int}(\cdot)$ iteratively over each concept $c \in C$ and for each agent $i \in \{1, ..., N\}$, we can construct a matrix $\boldsymbol{X}$ wherein each row is the outcome of an intervention:

$$\boldsymbol{X} = \begin{bmatrix} \text{Int}(\hat{c}_j, \bar{c}_j, 1) \\ \text{Int}(\hat{c}_k, \bar{c}_k, 2) \\ \vdots \\ \text{Int}(\hat{c}_l, \bar{c}_l, N) \end{bmatrix} = \begin{bmatrix} r^1_{\hat{c}_j} & r^2_{\hat{c}_j} & ... & r^n_{\hat{c}_j} \\ r^1_{\hat{c}_k} & r^2_{\hat{c}_k} & ... & r^n_{\hat{c}_k} \\ & \vdots & & \\ r^1_{\hat{c}_l} & r^2_{\hat{c}_l} & ... & r^n_{\hat{c}_l} \end{bmatrix}$$

[1] We carefully examine the possibility of OOD examples and propose ways to test this in Appendix C.4.

[2] Outcomes can also be defined in terms of other features (e.g., resources collected, agent proximity, etc.)



a) Two-phase Environment Overview. b) Concept Bottleneck Policies. c) Intervention Payoff Matrix. d) Sparse Intervention Graph.

Figure 2: **a)** A two-phase game. In phase one, Agent 1 and 2 choose from two levers (red, blue) using a special observation (wall color) and receive individual rewards for selecting the correct lever (blue) in the correct sequence (Agent 1 moves first, then Agent 2). If done correctly, the agents move on to the second phase. In phase two, Agent 1 selects from K doors, also using a special observation (gold star), and receives an individual reward for selecting the correct door. **b)** Both agents use a CBP: Agent 1 has concepts for both lever color and the correct door, while Agent 2's estimates lever color alone. **c)** It is possible to compute each agent's expected payoff under concept intervention. **d)** The graph obtained by running Equation (2) over pairs of concept interventions from the payoff matrix. Each edge is a statistical relationship between two interventions (color = edge weight)

We can then learn a graph that encodes the pairwise relationships between interventions by performing Lasso neighborhood selection [28] over $\boldsymbol{X}$. Specifically, for the intervention $\text{Int}(\hat{c}_j, i)$, we solve:

$$\min_{\beta_{ij}} ||\boldsymbol{X}_{ij} - \boldsymbol{X}_{\setminus ij}\beta_{ij}||_2^2 + \alpha||\beta_{ij}||_1 \tag{2}$$

where $\boldsymbol{X}_{ij}$ is the outcome of $\text{Int}(\hat{c}_j, \bar{c}_j, i)$, $\boldsymbol{X}_{\setminus ij}$ represents the remaining $N \times |C|-1$ interventions, and α is an L_1-penalty, enforcing sparsity. Under this graphical interpretation, the coefficient vector $\boldsymbol{\beta}_{ij} \in \mathbb{R}^{n-1}$ is used to establish edges between intervention outcomes. Crucially, non-zero edge weights in $\boldsymbol{\beta}_{ij}$ mark the similarity of $\text{Int}(\hat{c}_j, i)$'s outcome to other interventions. Outcome similarities may hint at the nature of agent behavior. We provide a comprehensive overview of Lasso neighborhood selection in Appendix D.

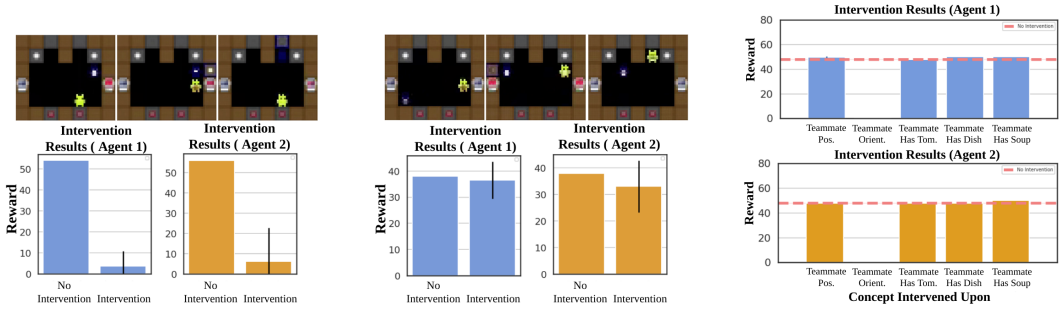
Illustrative Example: To build intuition, we present the following mathematical example. Consider the two-phase sequential game in Figure 2a. In the Phase 1 of this game, $N = 2$ agents spawn in a room that contains two levers (red and blue). One lever is set as a reward-giving lever (blue here) and is identified as such by an observable feature (wall color). Agent 1 acts first and must choose a lever to pull, receiving an individual reward ($r = 1.0$) for selecting the correct lever. Agent 2 then acts, also receiving $r = 1.0$ for selecting the correct lever. If both agents select the correct lever, the game proceeds to Phase 2; otherwise it is terminated. In Phase 2, Agent 1 spawns in a new room containing K unique doors (Agent 2 does not participate). Agent 1 must select a door to exit the room. One reward-giving door produces an individual reward ($r = 1.0$), while the other $K-1$ doors produce zero reward. As before, an observable feature (gold star) indicates which door is the correct one.

Next, assume both agents select actions with a CBP (see Figure 2b)—Agent 1's CBP conditions actions on two concepts: lever color and the door indicator; and Agent 2's CBP estimates lever color alone. If we assume that intervening over any concept estimate reduces decision-making to a random policy. we can compute the expected rewards for each agent under intervention, as shown in Figure 2c. Computing Equation (2) over these expected rewards with $K = 2$ returns a graph with a strong bi-directional edge between Agent 1's door color concept and Agent 2's lever color concept (see Figure 2d). This edge reveals an important inter-agent relationship: Agent 1's door selection in Phase 2 relies heavily on Agent 2's lever selection in Phase 1. In contrast, Agent 1's lever selection in Phase 1 does not depend on Agent 2 (evident by the lone self-loop for Agent 1's lever color intervention).

5 Experiments

In this section, we evaluate the extent to which our method addresses the following questions:

Identifying Emergent Coordination In cooperative settings, can concept intervention identify emergent coordination from policies that act independently, or how much coordination the environment



(a) Coordinating Agents. (b) Independent Agents. (c) Teammate Concepts.

Figure 3: Performance degradation under concept intervention uncovers (a) policies that coordinate and (b) policies that act independently. (c) For coordinating agents, intervening over teammate-related concepts exposes the factors driving the team's coordination strategy. Interestingly, teammate orientation is relied upon heavily by both agents for coordination.

demands? Can it expose lazy agents and, in general, measure an agent's contribution to the multi-agent system? Moreover, if agents *are* coordinating, what specific features underlie that coordination?
Emergent Strategic Behavior In competitive environments, do CBPs reveal strategic behaviors as they emerge during multi-agent self-play training? For example, can intervention help us better understand both inter-team emergent roles and intra-team counter-strategies that develop over time?
Inter-Agent Social Dynamics In mixed-incentive environments, how well does concept intervention identify inter-agent social dynamics? Can it expose the *functional connectivity* of the multi-agent system; or reveal chains of dependencies that might represent brittle inter-agent overfitting?

We use three environments from Melting Pot [21] for our experiments. Melting Pot environments vary along a number of multi-agent axes—incentive alignment (cooperative, mixed motive, adversarial), coordination requirements, etc—and therefore provide a strong benchmark for studying emergent behavior. Specific environments are described in the subsections below, with further details (including concept states) in Appendix B. For training, we use PPO [33], which has been shown to be state-of-the-art for multi-agent settings [9, 49], and train in a fully-decentralized fashion. We augment the PPO objective with our concept loss (called ConceptPPO moving forward). Additional training details and hyperparameter sweeps are provided in Appendix C.

5.1 Cooperative: Identifying Emergent Coordination

We first evaluate CBPs as a tool for understanding emergent coordination in fully-cooperative settings. We use the game Collaborative Cooking, where a group of agents inhabit a kitchen-like environment and must collaborate to find ingredients (tomatoes), complete recipes (bringing tomatoes to and from cooking pots), and deliver food as quickly as possible. We train 10 ConceptPPO policies with a concept cost weight of $\lambda = 0.1$ in this environment and use them in our evaluation below.

Do agents learn to coordinate? We aim to distinguish policies that learn coordination from those that solve the task independently. We evaluate each of the trained ConceptPPO policies over 100 test-time trajectories (and 5 seeds each) while intervening over all of the concepts related to each agent's teammate. The average cumulative reward under intervention is shown in Figure 3.

Figure 3 investigates a cooking environment with duplicate sets of ingredients (tomatoes) and tools (bowls, pots) on both sides. Therefore, both independent strategies and highly coordinated strategies can complete the task. In the trajectory in Figure 3a, we see an emergent strategy in which two agents coordinate. The orange agent works the bottom of the environment picking up tomatoes and bringing them to the cooking pot, while the blue agent stays in the top running dishes to and from the pot to deliver soup. Intervention over teammate-related concepts leads to a catastrophic drop in reward, as the agent's coordination clearly hinges on an accurate modeling of its teammate. In Figure 3b, a different strategy emerges, where agents complete the task independently on opposite sides of the kitchen. Here, intervention does not hurt reward, as neither agent needs to closely monitor its teammate to complete the task. This result therefore indicates that coordination detection through intervention is accurate and reliable with CBPs. In Appendix C.3, we study several cooking

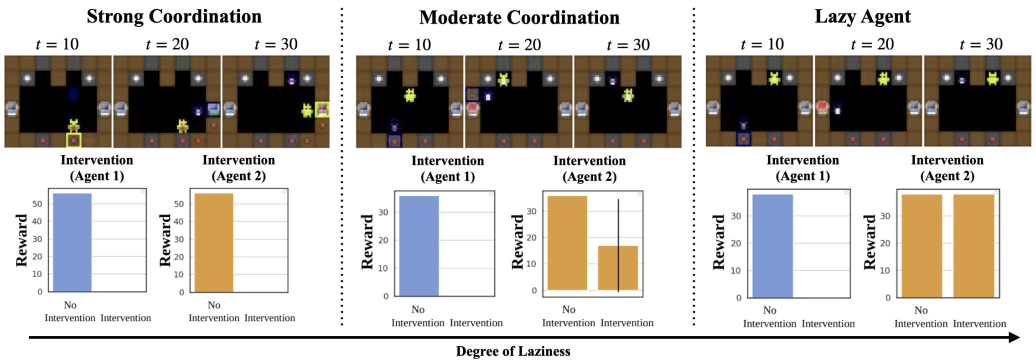


Figure 4: Intervening on an agent's concepts pertaining to itself exposes its *degree of laziness*.

environment layouts and average the reduction in reward observed when intervening on teammate concepts across all policies. We can thus measure the coordination demands of each environment.

How do they coordinate? Next, we examine the policies that *have* learned to coordinate and pinpoint the specific inter-agent features that drive their coordination (Figure 3c). Rather than intervening over all teammate-related concepts at once, we intervene one at a time. If agents are coordinating using a particular signal, the corresponding concept intervention should result in a sharp drop in performance.

Interestingly, performance only drops when intervening over the orientation concept. This suggests that agents are primarily using the orientation of other agents as a coordination signal, which is curious, as we might expect coordination to involve multiple sources of inter-agent information. For completeness, we conduct two supporting analyses. First, we rule out the possibility that intervening with a fixed orientation creates an OOD (or adversarial) input. We do this using an empirical sample of agent orientations to manufacture interventions that are both in- and out-of-distribution; and show that our results are consistent in both cases (see Appendix C.4.1). Second, we re-run this intervention over additional ConceptPPO policies trained without orientation as a concept. These results show that agent coordination latches onto yet another concept (teammate has_soup) as its primary signal (see Appendix C.4.2). The fact that agents learn a policy in which coordination hinges only on one particular signal reveals the brittleness of deep multi-agent reinforcement learning policies and their tendency to rely on highly specialized conventions. This brittleness is one of the reasons that MARL agents often fail to generalize to novel partners, or have poor zero-shot coordination performance [16].

Identifying Lazy Agents Our method also allows us to test for *coordination failures*. Here we test for lazy agents by replacing each agent's concept estimates about itself, including its own position, orientation, etc. If an agent is acting productively in the environment, removing this information will greatly hinder its performance; and team performance as a whole. If performance does not degrade, the agent likely is not contributing to the task. Figure 4 shows the results of this test for three policies, each differing in the strength of their contribution to the team.

The magnitude of performance degradation is a direct function of the productivity of each agent. When agents are both contributing (left), concept intervention has a strong negative impact on the performance of the agents. In the extreme lazy agent case where one agent does not move over the course of a trajectory (right), reward is not impacted by intervention. Interestingly, this intervention also captures intermediate effects. For example, when only one agent is productive, but both agents share a workspace, there is a small but noticeable drop in performance (middle). Thus, these results not only demonstrate that CBPs give us a unique way of diagnosing lazy agents, but also shows that it can quantify the *degree of laziness* of each agent as a function of performance degradation.

5.2 Emergence of Strategic Behavior

There, we intervene over orientation (i.e. R1's estimate of every other agent's orientation individually) and flag position (i.e. R1's estimate of the red flag and blue flag positions). These interventions give us insight into how important agent vs. environmental features are to the red agents' policies. The plots show the reward obtained by the red team under each of these interventions. The key takeaway is that the pattern of intervention outcomes reveals the roles that both agents have learned. R1 is negatively impacted by interventions over its own orientation, the blue team's attacking agent (B1), and its home flag. This pattern is consistent with the behavior of a base defender. R2 is negatively

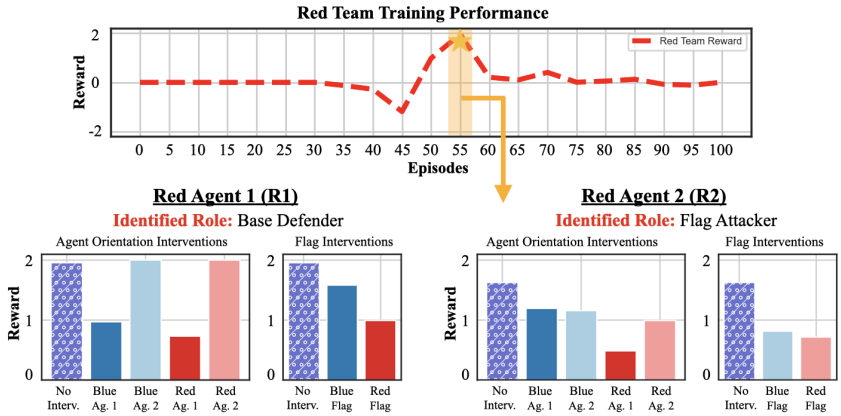


Figure 5: Emergent strategy in Capture the Flag. The red team discovers a dominant strategy at episode 55. For this episode, we intervene over orientation (e.g., R1's estimate of every other agent's orientation individually) and flag position (e.g., R1's estimate of both flag positions). These interventions give us insight into how important agent vs. environmental features are to the red agents. Specifically, they reveal two emergent roles: (i) Red agent R1 is identified as a flag defender agent—it models B1 (the blue attacking agent) and its home flag and is only negatively impacted by interventions over those concepts; (ii) R2 interacts with both blue agents, both flags, and is strongly impacted by intervention. R2 is thus identified as a flag attacking agent.

impacted by interventions over both flags and each of the other agents. This means that R2 is a flag attacker, navigating between the opponent base to take flags and its home base to return them.

Mixed cooperative-competitive environments induce an automatic skill learning curriculum for MARL agents similar to that of self-play [36], which can lead to highly-sophisticated behaviors such as the emergent tool use observed in team hide-and-seek by Baker et al. [3]. Here we investigate the extent to which CBPs can expose these strategic behaviors during training. We train independent CBP agents in Melting Pot's Capture the Flag environment (outlined further in Appendix B.4) and intermittently perform interventions over agent concept estimates during training. As before, we measure the magnitude of reward degradation.

Results from this analysis are shown in Figure 5. The reward curve represents one oscillation of self-play strategy for the red team—the red team receives negative reward in episode 45, but eventually counters with a dominant strategy in episode 55. Oscillations such as this are indicative of a newly learned strategic behavior by one or both agents [3]. But what strategic behavior was learned?

Conducting intervention anlysis at episode 55 reveals more precisely how the agents behave beyond red agents capturing the flag. Analysis of the first red agent (R1) shows that it is negatively impacted by interventions over itself, the blue team's attacking agent (B1), and its home flag. This pattern is consistent with the behavior of a *base defender*. Interventions over the second red agent (R2) expose a different pattern—the agent is negatively impacted by interventions over both flags and each of the other agents. This pattern suggests that R2 is a *flag attacker*, navigating between the opponent base to take flags and its home base to return them. A qualitative analysis of the agents' behavior is performed in Appendix C.9 and confirms these results. Thus, our intervention technique can be used *during training* to augment reward-based analysis and investigate strategic behaviors *as they emerge*.

5.3 Social Dilemmas: Inter-agent Social Dynamics

Social dilemmas further extend the scope of emergent behavioral complexity to include exploitation, free-riding, and public resource sharing. We evaluate our method's ability to uncover such interactions in Melting Pot's Clean Up, an environment in which agents must balance selfish behavior (harvesting fruit) with public service (cleaning a river is necessary for fruit to grow).

As we've seen, a natural way to investigate inter-agent dynamics with CPBs is by looking at outcomes with and without intervention. In Figure 6, we do the same for each agent in Clean Up and find that the rewards of Agent 1 and 2 are correlated under intervention. We also see an increase in Agent 1 and 2's idleness and a decrease in their inter-agent distance. A reasonable conclusion, therefore, is

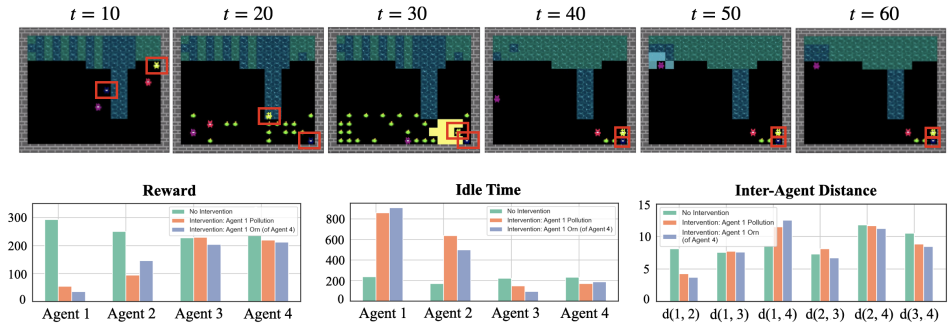
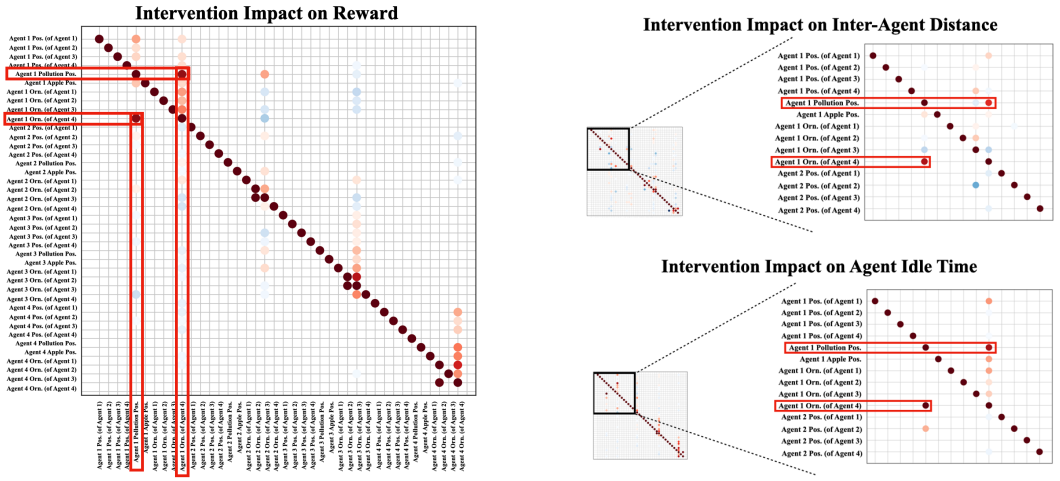


Figure 6: Learned social dynamics in Clean Up. Interventions with respect to reward and key behavioral features (agent idle time, inter-agent distance) an interesting relationship between Agents 1 (Blue) and 2 (Yellow). Both agents' interventions cause their rewards to decrease, idle time to increase, and distance to decrease, suggesting that Agent 1 and 2 collide as a result of these interventions (the qualitative result at the top confirms this).



(a) Graph over average reward features.

(b) Inter-agent distance and agent idleness graphs.

Figure 7: Social interactions in Clean Up. Each graph shows the result of running Equation (2) over all agent-concept pairs. (a) Intervening with respect to agent reward shows prominent bi-directional edges between Agent 1's estimate of the closest pollution and Agent 1's estimate of Agent 4's orientation (red highlights). (b) Intervening with respect to behavioral features results in the same prominent edges.The correlation between Agent 1 and 2's rewards, idle time, and distance (their collision in Figure 6) is therefore not the result of interrupting their coordination. Rather, it is the result of a chain of events that begin with a perturbation of agent 1's perception. Thus, our graph learning strategy has uncovered a chain of events that links the emergent behaviors of each agent).

that Agent 1 and Agent 2 are coordinating spatially and require accurate estimations of each other to complete the task. However, as we'll show, our method can uncover a more descriptive *chain of social interactions* that lead to these intervention outcomes. The trajectory in Figure 6 confirms this.

To investigate, we learn a series of graphs by computing Equation (2) over all agent-concept interventions using average reward, pairwise inter-agent distance, and the number of idle (non-moving) steps as features, respectively. Figure 7a shows the graph computed over rewards (in matrix form), which exposes a number of intriguing concept relationships across agents. First, there are *no* identifiable relationships between Agent 1 and 2 (no edges link interventions over Agent 1's concepts pertaining to Agent 2 or vice versa). This means that Agent 1 and 2 are in fact not relying directly on each other's concepts. Next, we find a strong bi-directional relationship between Agent 1's closest pollution concept and Agent 1's estimate of Agent 4's orientation. Importantly, this same bi-directional edge also exists in the graphs computed over inter-agent distance and idleness (see Figure 7b).

The graphs therefore reveal the following: First, the lack of a link between the Agent 1 and 2 in the graph suggests that their relationship is not one of coordination, but rather the result of an incidental

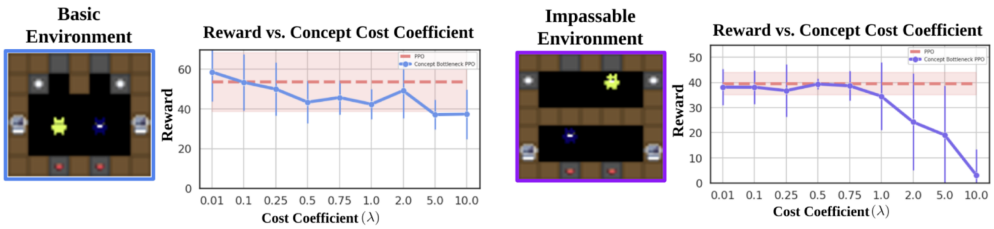


Figure 8: Asymptotic reward of ConceptPPO vs. PPO. For $\lambda \leq 0.5$, ConceptPPO matches the performance of non-concept-based PPO.

interaction in the environment (they are not explicitly modeling each other). Next, the bi-directional edge reveals that Agent 1's performance is largely dependent on its exploitation of the public service of Agent 4. Specifically, Agent 4 runs to and from the river to clean, at which point apples appear in the area of the patch that Agent 4 left. Agent 1 uses both Agent 4's orientation and its estimate of pollution (which Agent 4 changes) to decide when it should run to the area of the patch left by Agent 4 to consume apples. Because Agent 1's policy is so reliant on these two concepts, intervening over either of them causes Agent 1's behavior to collapse; and this happens in such a way that it interrupts Agent 2. Concretely, intervening over both Agent 1's pollution concept and Agent 1's estimate of Agent 4 lead to the same outcome: a collision between Agent 1 and Agent 2 (see Figure 7b).

In sum, the chain of dependencies is not Agent 1 $\leftrightarrow$ Agent 2, but rather Agent 1 $\rightarrow$ Agent 4 / Pollution $\rightarrow$ Agent 2 and rather than coordinating, we have discovered that Agent 1 and Agent 2 have overfit to brittle behavioral patterns that are easily disrupted. This is in turn why there is a correlation between their rewards under intervention. Our graph learning technique over CBP interventions thus reveals complex inter-agent dynamics that could take humans many hours to detect.

5.4 Bottleneck Performance

To evaluate our method more generally, we compare the asymptotic performance of CBPs relative traditional, non-concept-based PPO. We emphasize that PPO has been shown to be a state-of-the-art algorithm for decentralized multi-agent learning [49]. We measure performance as the average cumulative reward obtained over 100 test-time trajectories (and five random seeds each). Figure 8 provides an overview of these results for two Collaborative Cooking environments. There we find evidence that CBPs can match the performance of PPO across each of our environments for small values of the concept cost coefficient ($\lambda \leq 0.5$). This is an important result from the perspective of interpretability. It demonstrates that, if λ is tuned appropriately, it is possible to train intrinsically-interpretable policy networks—where, notably, decisions are expressed in human-understandable concepts—without sacrificing in task performance. Importantly, it also demonstrates the sufficiency of the concept set. Figure 8, and specifically the results for the Impassable Environment, show that over-valuing concept prediction loss causes performance to degrade. Performance falls for $\lambda > 0.75$ and, in all but the basic environment, collapses to zero for larger values ($\lambda \geq 5.0$). This breakdown occurs because, for high λ, the total gradient from L_C ($\nabla L_{RL} + \lambda \nabla L_C$) is dominated by the gradient from the concept-based loss L_C. In this case, gradient descent is not able to move the policy network's parameters in the direction of ∇L_{RL}, preventing policy optimization. We present our complete results across all environments in Appendix C.7.

6 Conclusion

We introduced Concept Bottleneck Policies as an interpretable, concept-based policy learning method for MARL and demonstrated that they are effective for understanding emergent multi-agent behavior. In particular, CBPs support concept intervention, which can be used to identify when a multi-agent team has learned to coordinate, what inter-agent features drive that coordination, and to what extent coordination is required in an environment. Moreover, concept intervention helps expose coordination failures like lazy agents and complex inter-agent dynamics. We discuss broader impacts, limitations, and future work in Appendix A.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[2] Dan Amir and Ofra Amir. Highlights: Summarizing agent behavior to people. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1168–1176, 2018.

[3] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

[4] Osbert Bastani, Carolyn Kim, and Hamsa Bastani. Interpretability via model extraction. *arXiv preprint arXiv:1706.09773*, 2017.

[5] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–30078, 2020.

[6] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[7] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

[8] Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, 2020.

[9] 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.

[10] Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

[11] Jessica Zosa Forde, Charles Lovering, George Konidaris, Ellie Pavlick, and Michael L Littman. Where, when & which concepts does alphazero learn? lessons from the game of hex. In *AAAI Workshop on Reinforcement Learning in Games*, volume 2, 2022.

[12] Ghost Town Games. Overcooked. https://store.steampowered.com/app/ 448510/Overcooked/, 2016.

[13] Asma Ghandeharioun, Been Kim, Chun-Liang Li, Brendan Jou, Brian Eoff, and Rosalind W Picard. Dissect: Disentangled simultaneous explanations via concept traversals. *arXiv preprint arXiv:2105.15164*, 2021.

[14] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. *Advances in Neural Information Processing Systems*, 32, 2019.

[15] Bradley Hayes and Julie A Shah. Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 303–312. IEEE, 2017.

[16] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pages 4399–4410. PMLR, 2020.

[17] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

[18] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[19] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pages 5338–5348. PMLR, 2020.

[20] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037*, 2017.

[21] Joel Z Leibo, Edgar A Dueñez-Guzman, Alexander Vezhnevets, John P Agapiou, Peter Sunehag, Raphael Koster, Jayd Matyas, Charlie Beattie, Igor Mordatch, and Thore Graepel. Scalable evaluation of multi-agent reinforcement learning with melting pot. In *International Conference on Machine Learning*, pages 6187–6199. PMLR, 2021.

[22] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[23] Siqi Liu, Guy Lever, Zhe Wang, Josh Merel, S. M. Ali Eslami, Daniel Hennes, Wojciech M. Czarnecki, Yuval Tassa, Shayegan Omidshafiei, Abbas Abdolmaleki, Noah Y. Siegel, Leonard Hasenclever, Luke Marris, Saran Tunyasuvunakool, H. Francis Song, Markus Wulfmeier, Paul Muller, Tuomas Haarnoja, Brendan Tracey, Karl Tuyls, Thore Graepel, and Nicolas Heess. From motor control to team play in simulated humanoid football. *Science Robotics*, 7(69): eabo0235, 2022. doi: 10.1126/scirobotics.abo0235.

[24] Siqi Liu, Guy Lever, Zhe Wang, Josh Merel, SM Ali Eslami, Daniel Hennes, Wojciech M Czarnecki, Yuval Tassa, Shayegan Omidshafiei, Abbas Abdolmaleki, et al. From motor control to team play in simulated humanoid football. *Science Robotics*, 7(69):eabo0235, 2022.

[25] Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. Explainable reinforcement learning through a causal lens. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2493–2500, 2020.

[26] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems*, 32, 2019.

[27] Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in alphazero. *arXiv preprint arXiv:2111.09259*, 2021.

[28] Nicolai Meinshausen and Peter Bühlmann. High-dimensional graphs and variable selection with the lasso. *The annals of statistics*, 34(3):1436–1462, 2006.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[30] Shayegan Omidshafiei, Andrei Kapishnikov, Yannick Assogba, Lucas Dixon, and Been Kim. Beyond rewards: a hierarchical perspective on offline multiagent behavioral analysis. *arXiv preprint arXiv:2206.09046*, 2022.

[31] Judea Pearl. Causal inference in statistics: An overview. 2009.

[32] Julien Perolat, Bart de Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. Mastering the game of stratego with model-free multiagent reinforcement learning. *arXiv preprint arXiv:2206.15378*, 2022.

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

[34] Pedro Sequeira and Melinda Gervasio. Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. *Artificial Intelligence*, 288:103367, 2020.

[35] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[36] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[37] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[38] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[39] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[40] Wolfgang Stammer, Patrick Schramowski, and Kristian Kersting. Right for the right concept: Revising neuro-symbolic concepts by interacting with their explanations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3619–3629, 2021.

[41] 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.

[42] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[43] Eugene Vinitsky, Nathan Lichtlé, Kanaad Parvate, and Alexandre Bayen. Optimizing mixed autonomy traffic flow with decentralized autonomous vehicles and multi-agent reinforcement learning. *ACM Transactions on Cyber-Physical Systems*, 7(2):1–22, 2023.

[44] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

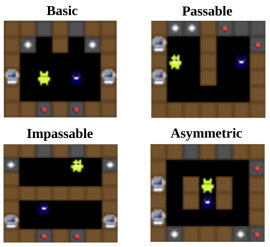
[45] Cathy Wu, Abdul Rahman Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: A modular learning framework for mixed autonomy traffic. *IEEE Transactions on Robotics*, 38(2):1270–1286, 2021.

[46] Sarah A Wu, Rose E Wang, James A Evans, Joshua B Tenenbaum, David C Parkes, and Max Kleiman-Weiner. Too many cooks: Bayesian inference for coordinating multi-agent collaboration. *Topics in Cognitive Science*, 13(2):414–432, 2021.

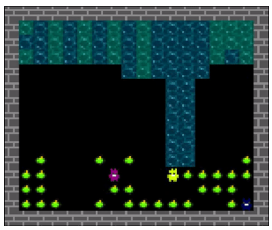
[47] Zhao Yang, Song Bai, Li Zhang, and Philip HS Torr. Learn to interpret atari agents. *arXiv preprint arXiv:1812.11276*, 2018.

[48] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems*, 33:20554–20565, 2020.

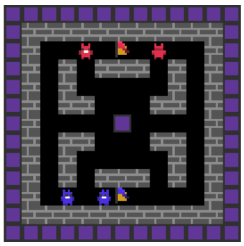
[49] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.



(a) Collaborative Cooking.



(b) Clean Up.



(c) Capture the Flag.

Figure 9: Environment Overview.

A Discussion

Limitations One of the limitations of the proposed method is that it requires manual specification of human-interpretable concepts to be used in the policy bottlenecks at training time. As illustrated in the experiments section of the paper, these are often intuitive characteristics of the specific environment targeted (e.g., positions and orientations of agents, states of various objects that can be interacted with, etc.), which can be then automatically extracted from the environment simulator. In future work, it would be interesting to learn such key relevant concepts automatically. Another potential limitation of the method is that the combination of traditional reward-based RL loss and concept-loss can make for a slightly complex loss landscape. However, as illustrated by our experiments, one mechanism for controlling this is via the concept loss λ parameter. To make the selection of this parameter easier, a baseline of the agent policies ran *without* the concept bottlenecks can be used to establish reasonable bounds on their expected performance in the task. Subsequently, the parameter can be swept over (similar to our experiments) to identify a λ that leads to accurate concept learning with minimal impact on final performance. Finally, we note that we do not provide theoretical guarantees that CBPs converge to a specific solution concept (i.e. refinement of the Nash equilibrium). In general, such guarantees are difficult to provide when combining MARL with neural networks and the addition of a concept-based loss further complicates this picture; though we do point out that the lack of such guarantees in many prior MARL works highlights the importance of the interpretability problem.

Broader Impacts In terms of potential societal implications, modeling of interactive decision-making and concept estimations of agents in such systems might be used by adversarial actors to intervene on concepts and intentionally cause performance degradation or agents misbehaving. On the other hand, concept bottlenecks themselves could be a means of identifying such adversarial attacks, by detecting shifts in concept estimations that were otherwise unexpected.

Future Work There are a number of interesting avenues for future work in the area of concept bottlenecks for MARL. First, taking inspiration from the literature on POMDPs and belief-space planning, we can explore extensions to our architecture where agents can leverage both histories of observations and histories of concepts in training and concept estimation. This may also be combined with architecture improvements, such as giving agents memory (e.g., through recurrent policies). Further, there is room to explore the use of concept prediction accuracy as an intrinsic reward that incentivizes agents to explore subsets of the state space in which they do not predict concepts accurately (thereby improving their concept estimates). An important future work is to extend our simple graph-based analysis technique to more complex graph learning paradigms [10].

B Additional Environment Details

B.1 Setup

The following details are common across all of our environments:

- *States*: Each environment is a grid world. Grid cells can be filled by an agent or an environment-specific object.

- *Observations*: Agents receive partial multi-modal observations consisting of their own position and orientation in the grid, as well as a partial RGB rendering of a 5-cell $\times$ 5-cell window centered at the agent.
- *Actions*: Agents can execute one of 8 actions: no-op, move $\{\texttt{up}, \texttt{down}, \texttt{left}, \texttt{right}\}$, turn $\{\texttt{left}, \texttt{right}\}$, and interact.

B.2 Collaborative Cooking

A visualization of the four cooking environments from our experiments are shown in Figure 9a. Based on the game Overcooked [12] and subsequent work that has developed Overcooked-like environments for studying multi-agent coordination [7, 46], Melting Pot's Collaborative Cooking is a game in which a group of agents inhabit a kitchen-like environment and must collaborate to find ingredients, complete recipes, and deliver finished dishes as quickly as possible. Solving the cooking task requires sophisticated coordination, involving both task partitioning—splitting a recipe into parts—and role assignment—distributing sub-tasks among agents. For these reasons, Collaborative Cooking is investigated in a number of prior works [46, 7, 41] and is emerging as a strong benchmark for multi-agent learning. The Collaborative Cooking game for N agents is defined as follows:

- *Recipe*: (i) Bring a tomato to a cooking pot (3 times), (ii) Wait for soup to cook in the pot (20 time-steps), (iii) Bring a dish to the cooking pot, (iv) Pour soup from the pot into the dish, (v) Deliver soup to the delivery location. In practice, solving this task from scratch in its entirety is extremely difficult, as each of the aforementioned steps requires agents to execute a series of movement and interaction actions in sequence.
- *States*: Grid cells can be filled by an agent or any of the following items: Floor, Counter, Cooking Pot, Dish, Tomato.
- *Reward*: By default, agents share a positive reward for completing the entire recipe outlined above. In practice, however, solving the cooking task with this sparse reward alone is infeasible (completing each of the recipe steps through random exploration is prohibitively challenging) and successful approaches in prior works either pair learning agents with helpful bot agents or introduce "densified" pseudorewards to augment the agents' learning signal [20]. We implement the latter, giving agents a small positive reward for completing steps (i) and (iii) of the recipe. More concretely, we define the following three-part reward:

$$r_t = \begin{cases} 20, & \text{if soup cooked and delivered.} \\ 1, & \text{if tomato placed in cooking pot.} \\ 1, & \text{if soup poured into dish.} \\ 0, & \text{otherwise.} \end{cases}$$

- *Concepts*: We assume that each environment supports the following concepts (and concept types): (i) agent position (scalar); (ii) agent orientation (scalar); (iii) whether or not an agent has a tomato, dish or soup (binary); (iv) cooking pot position (scalar) (v) the progress of the cooking pot (scalar); (vi) the number of tomatoes in the cooking pot (categorical); and (vii) the position of each tomato and dish (scalar).

B.3 Clean Up

The Clean Up game is shown in Figure 9b. Clean Up is a classic social dilemma in which agents are forced to balance selfish behaviors with public goods. Each agent is rewarded proportionally to the number of apples it harvests. However, the rate at which apples respawn after they are eaten is directly related to the amount of pollution that exists in the river. Over time, pollution builds up in the river if it remains uncleaned until eventually, apples are prevented from growing altogether. Agents must learn to periodically clean the river to ensure that apples continuously grown, even though they are not directly incentivized to do so by default. Therefore, it is possible for agents to develop both emergent "free-loading" behaviors—harvesting apples without cleaning the river—and emergent public service behaviors—cleaning the river while trying to harvest as many of the remaining apples after cleaning. This makes Clean Up a fitting environment with which to study the extent to which inter-agent social dynamcis can be revealed. Some environment-specific details are outlined below:

- *States*: Grid cells can be filled by an agent or any of the following items: Floor, Wall, Apple, Clean River, Polluted River.
- *Reward*: Agents receive an individual positive reward for collecting apples. By default, agents are not rewarded for cleaning the river. We found that this greatly increased the amount of training time required for agents to learn to clean the river (because they have no incentive to do so). Because we are motivated by understanding social behaviors once they have emerged, we augmented each agent's reward with a small positive reward for cleaning the river. Specifically, we define the following reward for the cleaning task:

$$r_t = \begin{cases} 1, & \text{for eating an apple.} \\ 0.01, & \text{for cleaning pollution.} \\ 0, & \text{otherwise.} \end{cases}$$

- *Concepts*: We assume that each environment supports the following concepts (and concept types): (i) agent position (scalar); (ii) agent orientation (scalar); (iii) closest pollution position (scalar); (iv) closest apple position (scalar).

B.4 Capture the Flag

The Capture the Flag game is shown in Figure 9c. Capture the Flag is a well-known competitive environment in the multi-agent literature [17]. The game pits two multi-agent teams (red, blue) head-to-head in an arena-style environment. The red team is spawned at the top of the environment, and the blue team is spawned at the bottom. Both team have at their "home base" a flag of their team's color. The goal of each team is to capture the opposing team's flag as many times as possible within the game's time horizon; while also preventing the opposing team from capturing the team's home flag. The primary defense mechanism for an agent is a "zap" action that reduces an opponent agent's health if it is in the zap radius, and forces the agent to drop the flag (if it is holding one). In addition to zapping other agents, agents can choose a "paint" action that colors the floor tiles with the painting agent's home color. Agents of the opposite color cannot move over painted tiles until they paint it their own color, so the painting mechanism can be used strategically to slow down the movement of opponents through certain corridors. There is also a set of purple tiles in the environment (one in the center, many surrounding the arena) that serves as an indicator of whether or not a flag has been taken from its home base. This tile serves as an observation for each agent, informing that agent of the current state of both team's flags (purple = no flags taken, blue = blue flag taken, red = red flag taken, gray = both flags taken).

Sophisticated Capture the Flag agents can learn a number of complex strategies, such as attacking an opponent's flag, battling opponent agents for territory, and base camping to protect the home flag. Agents are forced to learn these competitive behaviors from scratch and refine those behaviors during training. From the perspective of Concept Bottleneck Policies, therefore, Capture the Flag is a strong case study of the emergence of strategic behaviors over time. Some environment-specific details are outlined below:

- *States*: Grid cells can be filled by an agent or any of the following items: Wall, Red Flag, Blue Flag, Red Paint, Blue Paint, Flag Indicator Tile.
- *Reward*: Agents receive a shared team reward for capturing the opposing team's flag (by running it back to base). Agents are also rewarded individually for picking up a flag from the opposing team's base, returning a previously stolen flag to base, and zapping an opponent flag carrier. Agents are therefore most strongly motivated to act offensively, but must learn to balance attacks with defensive strategy. In sum, we define the following reward for the task:

$$r_t = \begin{cases} 1, & \text{for capturing the opponent's flag..} \\ 0.01, & \text{for picking up the opponent's flag.} \\ 0.01, & \text{for returning a flag to base.} \\ 0.01, & \text{for zapping an opponent flag carrier.} \\ 0, & \text{otherwise.} \end{cases}$$

- *Concepts*: We assume that each environment supports the following concepts (and concept types): (i) agent position (scalar); (ii) agent orientation (scalar); (iii) flag position (scalar);

Table 1: Hyperparameter sweeps for training ConceptPPO and PPO. Swept values are shown in braces and highest-performing values are bolded.

Hyperparameters	
Name	Value
Training Steps	25e6
Batch Size	$\{64, 128, 256, \mathbf{512}, 1024\}$
Learning Rate	$\{1e{-}3, \mathbf{1e{-}4}, 1e{-}5\}$
Gradient Norm	$\{0.1, \mathbf{0.5}, 1.0, 5.0, 10.\}$
PPO Unroll Length	$\{4, 8, \mathbf{16}, 32\}$
PPO Clipping ϵ	$\{0.01, \mathbf{0.05}, 0.1, 0.2, 0.3\}$
PPO Entropy Cost	$\{0.001, \mathbf{0.01}, 0.05\}$
PPO Value Cost	$\{0.75, 0.9, \mathbf{1.0}\}$

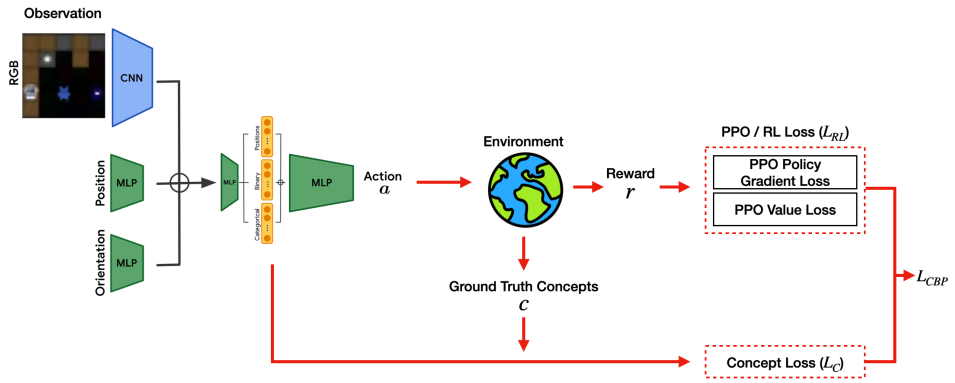


Figure 10: An architecture diagram showing how the concept bottleneck loss L_{CBP} is computed in L_C . Crucially, PPO is applied only over L_{RL}, which captures environmental rewards (e.g., delivering in collaborative cooking). Separately, concept loss L_C is computed in a supervised fashion using concept labels that are extracted from the environment. The concept loss is **not** incorporated into an agent's reward function and therefore does not incentivize its behavior directly.

(iv) whether or not an agent has the opponent's flag (binary) (scalar); (v) floor paint color (categorical); (vi) flag indicator tile color (categorical).

C Training Details

C.1 Architecture and Hyperparameters

For both PPO and ConceptPPO, each agent's policy network consists of CNN and MLP encoders (for image and position/orientation inputs, respectively), followed by a two-layer MLP and a linear mapping that compresses the encoded inputs into concept predictions. Concept estimates are fed through a two-layer MLP, which produces the final action. ReLU activation is used throughout (except in the bottleneck layer itself). As a baseline, we use vanilla PPO (no concept loss) with the same architecture. We train 10 individual policies across each of the following values of λ: $\{0.01, 0.1, 0.25, 0.5, 0.75, 1.0 2.0, 5.0, 10.0\}$. We conducted a wide hyperparameter sweep to train both ConceptPPO and PPO, which is summarized in Table 1.

C.2 Loss Breakdown

In this section, we present a more detailed discussion of the concept bottleneck loss introduced in Section 4 and, specifically, L_C. The objective L_{CBP} is a function of two separately computed objectives: (i) L_{RL}, which is a reward-based loss that operates only over environmental rewards (e.g., delivering in collaborative cooking); and (ii) L_C, which is a supervised loss computed over concept labels that are extracted from the environment. We are leveraging PPO to optimize each agent's

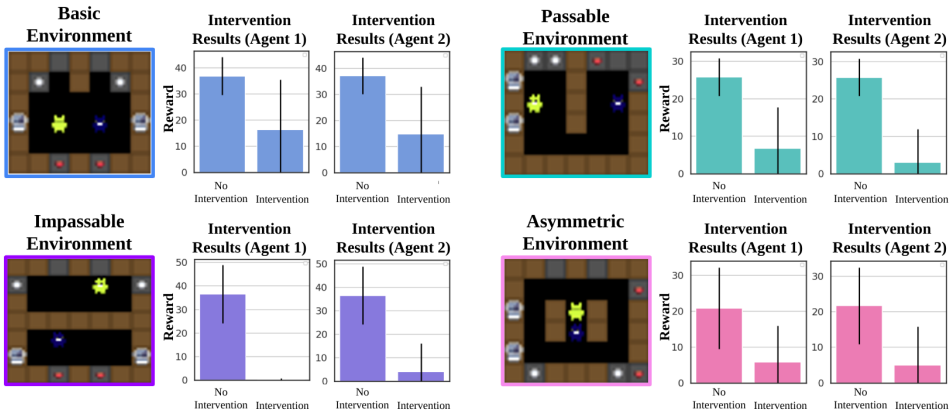


Figure 11: **Does the environment require coordination?** Averaging the impact of concept intervention over all policies trained in an environment reveals the extent to which coordination is required by that environment. In the impassable environment, agents cannot solve the task without coordinating, leading to consistent performance drops under intervention. In the basic environment, policies that coordinate (and therefore fail under intervention) are averaged with policies that act independently (and are uninterrupted by intervention), so the overall impact of intervention is less severe.

policy, which includes terms for both value error and generalized advantage estimation. A crucial detail of our architecture is that PPO's value and advantage estimation operates over environmental rewards only. The accuracy of concept predictions is not incorporated into an agent's reward in any way—i.e. we are not adding an auxiliary/intrinsic rewards associated with the concept predictions to the advantage function estimate. This is done intentionally so as not to bias agent behavior away from states that result in high concept error prediction.

C.3 Environmental Demands of Coordination

For each Concept PPO and PPO policy trained in our cooking environments, we mask out, for each agent, all of the concepts related to that agent's teammate. We measure the average cumulative reward attained over 100 trajectories each. The results of this intervention test are shown in Figure 11. The level of coordination required by the environment is apparent from the severity of performance degradation that results from intervening on each agent's estimates of its teammate. Most notable is the contrast between the basic environment (blue) and the impassable environment (purple). The impassable environment requires strict coordination—agents can only access a subset of ingredients and must pass items to each other across the center divider. In this case, intervention performance drops to near-zero across all policies. In the basic environment, on the other hand, there are no obstacles and agents have access to their own supply of ingredients; and so both policies in which agents coordinate and policies in which agents act independently are successful. Consequently, the impact of intervention is much less severe, as the performance of policies that coordinate (and fail under intervention) is averaged in with independent policies that are unaffected by intervention. Altogether, these results indicate that our method accurately distinguishes environments that require coordination from those that do not.

We also present results for this analysis in two additional cooking environments: passable and asymmetric. As in the basic and impassable environments, we find that the coordination demands of the asymmetric and passable environments can be revealed through concept intervention. Both environments involve moderate coordination—the most efficient strategy for bringing items to and from the cooking pot involves passing them over the counter from one agent to another—but this coordination is not required (like in the impassable environment). For this reason, we still see a significant drop in the performance of our agents under intervention, but not a full decrease to zero.

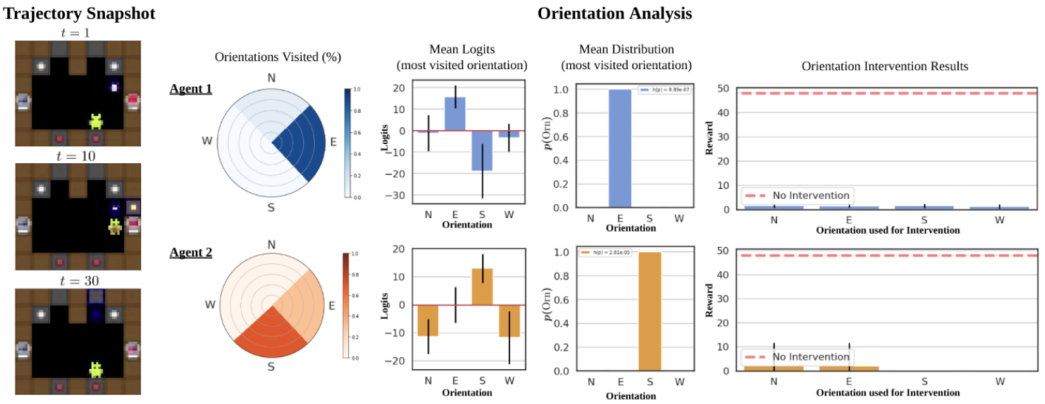


Figure 12: Overview of our method for constructing in-distribution concept masks for intervention. Using test-time trajectories, we compute an empirical distribution of the orientations experienced by each agent. Next, we compute the mean logits (and corresponding distribution) produced by each agent for for the most frequently visited orientation of its teammate. Finally, we report the results of intervening with this mean logits vector in place of each of the four cardinal orientations. The results of this intervention are consistent, regardless of whether the mask used was in- or out-of-distribution.

C.4 Factors of Coordination Analysis (cont'd)

C.4.1 In vs. Out-of-Distribution Orientation Analysis

Here we further examine the impact of intervening on orientation. First, we compute an empirical distribution of orientations that each agent vists over 100 test-time trajectories (across five random seeds each). From those trajectories, we compute both the mean logits vector produced by each agent's concept bottleneck for the *most frequently visited orientation*, and the distribution represented by those logits. Crucially, the mean logits vector for the most frequently visited orientation can be used as an in-distribution value for concept interventions—it is an orientation estimate that each agent has likely seen before. Similarly, we can create an out-of-distribution concept intervention mask by permuting the mean logits vector such that the probability mass shifts a different cardinal orientation. Using these manufactured orientations, we perform the same intervention technique as before, iteratively replacing each agent's orientation concept with the mean logits vector in place of each cardinal direction, and measure performance of the multi-agent team.

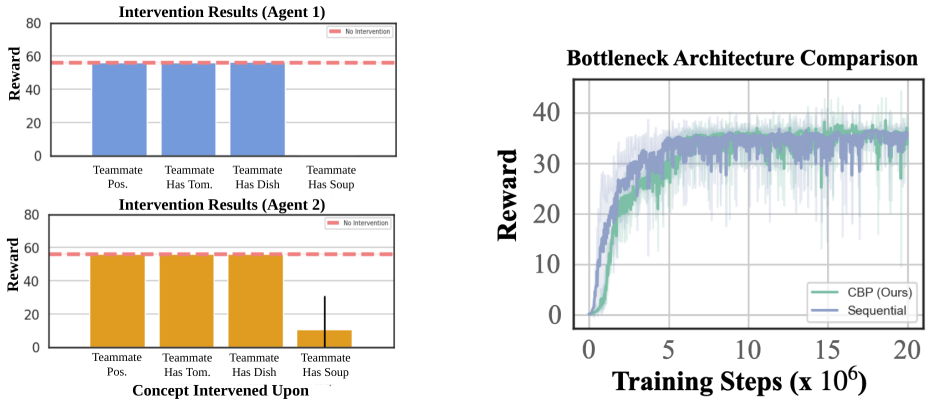
To provide intuition for this technique, we ground it in the cooking task used for our experiments. Consider, for example, the tomato-picking agent in the trajectory snapshot of Figure 12, who primarily faces south and east—the directions needed to pick tomatoes from the bottom counter and place them in the cooking pot on the right-hand side. The tomato-picking agent's teammate (the waiter agent) must learn to accuately model these orientations to satisfy its concept prediction objective, and so frequently passes low-entropy distributions for south and east to its policy network. Intervening on the waiter agent's orientation estimate with a low-entropy distribution for south and east, therefore, creates an in-distribution mask, whereas intervening with a low-entropy distribution for north or west creates an out-of-distribution mask.

The results of this intervention test are shown in Figure 12, alongside the orientation distributions, mean logits, and the distribution represented by those logits. Interestingly, the previously observed degradation of performance as a result of intervening on teammate orientation is upheld, regardless of whether the mask value is in- or out-of-distribution.

This provides further evidence that the agent's reliance on orientation is a legitimate artifact of their emergent strategy and not an adversarial or OOD example.

C.4.2 Intervention Without Orientation

To further investigate the surprising use of orientation as the primary signal driving the emergent behavior of our learning agents, we train a set of ConceptPPO policies in our basic environment without orientation as a concept (across each λ value as before). We then perform the same iterative



(a) Intervention without orientation.

(b) Bottleneck architecture test.

Figure 13: Additional bottleneck analyses. (a) Results of iterative concept intervention on Concept PPO agents trained without orientation. (b) Asymptotic performance of joint CBP architecture (ours) vs. sequential CBP architecture.

intervention analysis over the concepts pertaining to each agent's teammate (as outlined in Section 5.1 (excluding orientation, of course). The results of this analysis are shown in Figure 13a. After removing each agent's orientation concept, we find that the agents latch onto a new concept–"has soup"–as the primary concept that drives their coordination.

C.5 Bottleneck Architecture Comparison

Consider a concept bottleneck composed of two functions: (i) $g : x \to c$ mapping inputs to concept estimates; and (ii) $f : c \to y$ mapping concept estimates to outputs. The work of Koh et al. [19] compares three architectures for concept bottlenecks in supervised learning settings that lean these functions in different ways:

- **Independent Bottleneck**: This architecture learns functions $\hat{f}$ and $\hat{g}$ independently, where $\hat{g}$ is trained separately to predict concept estimates $\hat{c}$ from inputs x using ground truth concepts c as supervision. $\hat{f}$ is trained to predict outputs $\hat{y}$ from ground truth concepts c, using ground truth labels y as supervision.
- **Sequential Bottleneck**: This architecture learns $\hat{g}$ first, then uses concept predictions from $\hat{g}$ to train $\hat{f}$. Specifically, $\hat{g}$ is trained in the same manner as in the independent bottleneck. $\hat{f}$ is then trained to predict outputs $\hat{y}$ from concept estimates $\hat{c} = \hat{g}(x)$, using ground truth labels y as supervision.
- **Joint Bottleneck**: This architecture trains $\hat{f}$ and $\hat{g}$ together. The models are trained jointly to predict $\hat{y} = \hat{f}(\hat{g}(x))$ using an objective that is a weighted sum of loss terms for concept prediction (using ground truth concepts c as supervision) and target label prediction (using ground truth labels y as supervision).

Our proposed CBP architecture is most like the joint bottleneck. Here we compare the performance of our proposed architecture to a MARL equivalent of the sequential bottleneck. A simple way to obtain an approximation of the sequential bottleneck using the same architecture (shown in Figure 10) and objective (defined by L_C) as CBPs is by the stopping gradient flow between the concept estimator network and policy head during backpropagation.

We train both CBPs (our proposed joint bottleneck architecture) and sequential bottlenecks in the basic cooking environment shown in Figure 9a. Comparing the asymptotic reward achieved by both approaches shows that joint CBPs (our architecture) and the sequential bottleneck (implemented via stop gradient) perform comparably. This gives us confidence that the joint architecture is not unfairly surpassing the performance of other bottleneck architectures by "hacking" concept estimates. We investigate concept leakage more concretely in the following subsection.

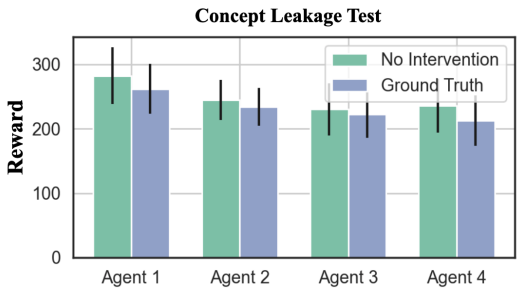


Figure 14: Concept leakage test. We perform an intervention over each agent's concept estimates with the ground truth concept labels from the environment. There is a very slight decrease in performance during intervention, which indicates that some concept leakage does exist. However, the small amount of performance degradation indicates that agents are faithfully learning to estimate concepts.

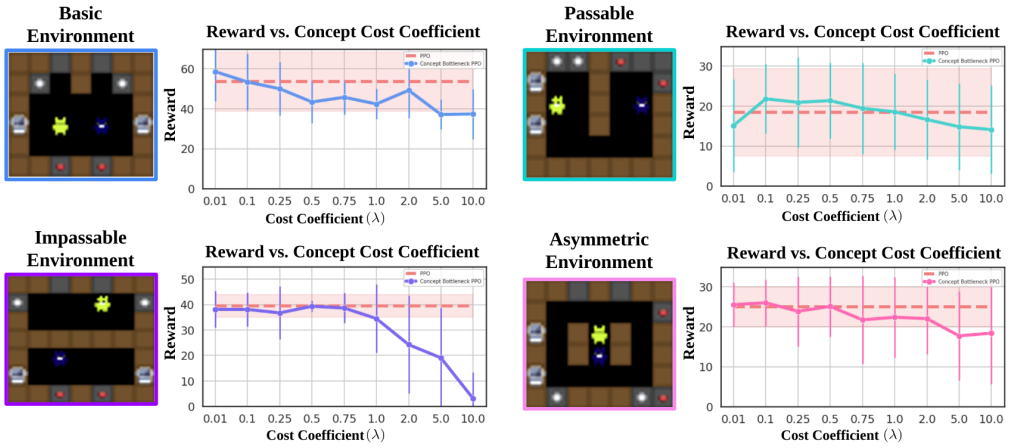


Figure 15: Asymptotic reward of ConceptPPO vs. PPO. For $\lambda \leq 0.5$, ConceptPPO matches the performance of non-concept-based PPO.

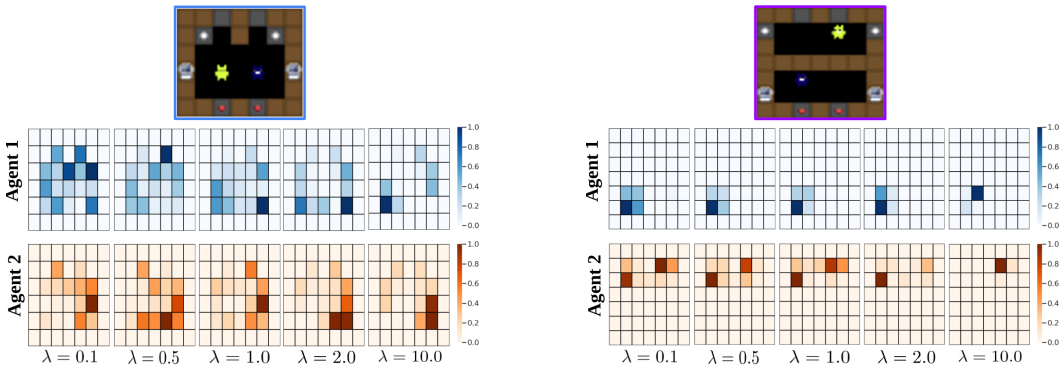
C.6 Concept Leakage Analysis

To examine the extent to which concept leakage occurs as a result of the joint objective outlined in L_C, we perform an evaluation of CBPs while intervening over each agent's concept estimates with ground truth concept values (for all concepts). Specifically, at each time-step, we replace each agent's concept estimates with the ground truth concept values extracted from the environment. Intuitively, if concept leakage is a rampant issue in our proposed architecture and agents are learning to "hack" their concept estimates to encode additional side channel information, we should see performance degrade significantly as a result of this intervention. We perform this analysis over 100 test-time trajectories (across 5 random seeds each) using the trained CBP policies from the Clean Up analysis in Section 5.3.

The results of this analysis are shown in Figure 14, which compares the average reward under the aforementioned ground truth intervention analysis to the average reward obtained by agents with no interventions (using concept estimates as usual). Interestingly, there is a slight decrease in performance that occurs as a result of the ground truth intervention, indicating that some concept leakage may be present in the agents' CBPs. However, the small magnitude of performance degradation indicates that agents are reliably encoding the true concept values in their bottleneck predictions.

C.7 Bottleneck Performance

To evaluate our method more generally, we compare the asymptotic performance of Concept PPO and PPO. We measure performance as the average cumulative reward obtained over 100 test-time trajectories (and five random seeds each). Figure 15 provides an overview of these results. We find



(a) Basic Environment.

(b) Impassable Environment.

Figure 16: State visitation results. As concept cost coefficient λ increases, behavior collapses.

evidence that ConceptPPO can match the performance of PPO across each of our environments for small values of the concept cost coefficient ($\lambda \leq 0.5$). This is an important result from the perspective of interpretability. It demonstrates that, if λ is tuned appropriately, it is possible to train intrinsically-interpretable policy networks—where, notably, decisions are expressed in human-understandable concepts—without sacrificing in task performance. Importantly, it also demonstrates the sufficiency of the concept set.

Figure 15 also shows that over-valuing concept prediction loss causes performance to degrade. Performance falls for $\lambda > 0.75$ and, in all but the basic environment, collapses to zero for larger values ($\lambda \geq 5.0$). This breakdown occurs because, for high λ, the total gradient from L_C ($\nabla L_{RL} + \lambda \nabla L_C$) is dominated by the gradient from the concept-based loss L_C. In this case, gradient descent is not able to move the policy network's parameters in the direction of ∇L_{RL}, preventing policy optimization.

C.8 State Visitation Analysis

To illustrate the behavioral changes induced by λ, we measure the distribution of states visited by each agent across a subset of the λ values used during training. The results for each agent are plotted as a heatmap in Figure 16. As λ increases, multi-agent behavior begins to break down, as the gradient signal from environmental reward gets over-shadowed by that of the concept prediction objective. In the most extreme cases (e.g., $\lambda = 10.0$ in the impassable environment) agents fail to make any progress in completing the task.

C.9 Role Assignment Results (Cont'd)

Full Temporal Analysis Here we supplement our analysis of role assignment in capture the flag from Section 5.2 by presenting the complete set of reward and intervention curves over time (Figure 5). As before, we hone in on interventions over agent and flag-related concept estimates from Agent 3 and Agent 4 (from the red team). Baseline performance with no intervention is shown as the red-dashed line. As shown in Figure 5, test-time reward is zero both with and without intervention in the early stages of training, as agents are first interacting with the environment. A first swing of reward occurs around checkpoint 45, where it appears that the blue team has learned to capture the red team's flag (reward for the red agents is negative). This is followed by a quick counter-swing in which the red team begins collecting positive reward. We will hone in on the intervention analysis at checkpoint 55, as it is the first time we see the red team learn productive behavior. At checkpoint 55, we see an interesting pattern in the intervention results for Agent 3 vs. Agent 4. Te team receives positive reward, so we know that the red team is capturing the blue team's flag, but the intervention analysis reveals more precisely how that is done. First, in the top row of Figure 17, intervening over Agent 3's orientation concepts shows that the red team is negatively impacted by inteventions over Agent 3's orientation estimate for Agent 1 (a blue agent) and Agent 3 (itself). Intervening over Agent 4 results in a slightly different pattern—every intervention over orientation concepts degrades the red teams performance! As we have seen previously, agents tend to model the other agents that they interact with the most, suggesting that Agent 4 is interacting with all of the other agents in the

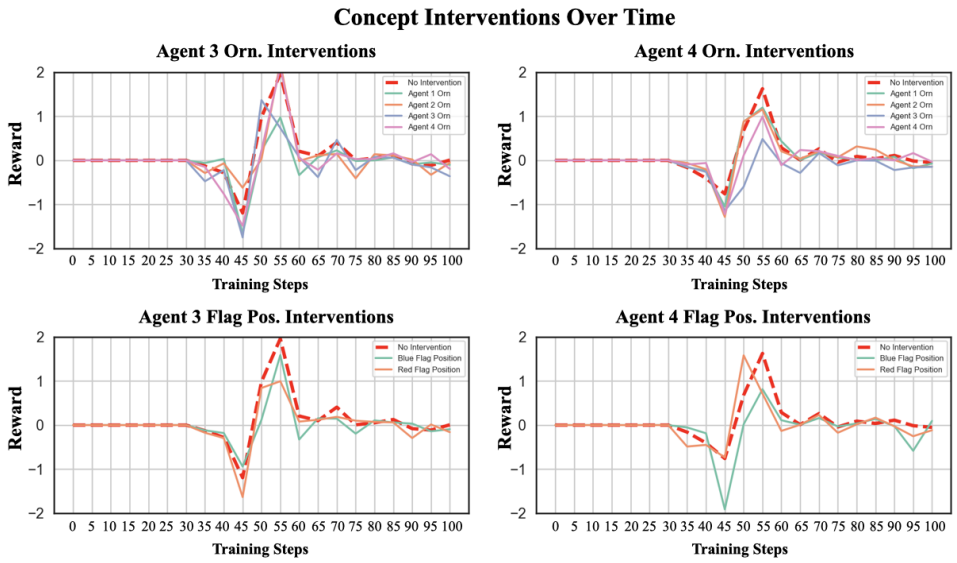


Figure 17: Concept interventions are performed intermittently over time for each agent. Results for the agents on the red team (Agent 3, Agent 4) are shown here. In the top row, we show results of an intervention over agent-related concepts (orientation), and in the bottom row we show results of an intervention over flag-related concepts (flag position). We find an interesting pattern in the intervention results at checkpoint 55. First, we find that Agent 4 is negatively impacted by all of the orientation interventions, whereas Agent 3 is only negatively impacted when intervening over the orientation of Agent 1 and Agent 3 (itself). Next, we find that Agent 4 is similarly negatively impacted by both flag interventions, whereas Agent 3 is only negatively impacted by an intervention over the red flag's position. The intervention reveals, therefore, that Agent 4 is likely an attacking agent and Agent 3 is likely a defending agent.

environment, whereas Agent 3 is interacting with one blue agent. We see a similar pattern in the flag-related interventions as well (bottom row of Figure 5. There we find that, at checkpoint 56, Agent 3 is negatively impacted by an intervention over the red flag's position, but not the blue flag's position. Agent 4, however, is negatively impacted by intervention over both flag positions. This further suggests that Agent 3 interacts primarily

Altogether, the clues that we get from this intervention analysis suggest the following: (i) First, Agent 4 interacts with each agent in the environment (spanning both teams) and also interacts with both flags. The intervention analysis reveals, therefore, that Agent 4 has learned an attacking strategy and is achieving positive reward by capturing the blue teams flag; (ii) Next, we see that Agent 3 interacts primarily with its own flag (the red flag) and also interacts with one of the blue agents. The intervention analysis reveals, therefore, that Agent 3 has learned a defensive strategy, and is likely maintaining the red team's positive reward by fending off a blue attacker.

Qualitative Results Figure 18 provides a snapshot of capture the flag behavior exhibited by the CBP agents' at episode 55. As predicted by our intervention analysis in Section 5.2, one agent (R2) has learned an attacking role—traversing to the blue team's base, capturing the flag, and returning it to the red base (and passing both blue agents along the way)—and the other agent (R1) takes on a defending role—camping at the red team's base and defending it from a blue attacker. Altogether, these results confirm that intervention can be used *during training* to augment reward-based analysis and investigate strategic behaviors *as they emerge*.

D Lasso Neighborhood Selection

Lasso neighborhood selection is a simple method that poses graph learning as a Lasso regression problem [10]. Let $\boldsymbol{X}$ be an observation matrix that is composed of some set of random variables

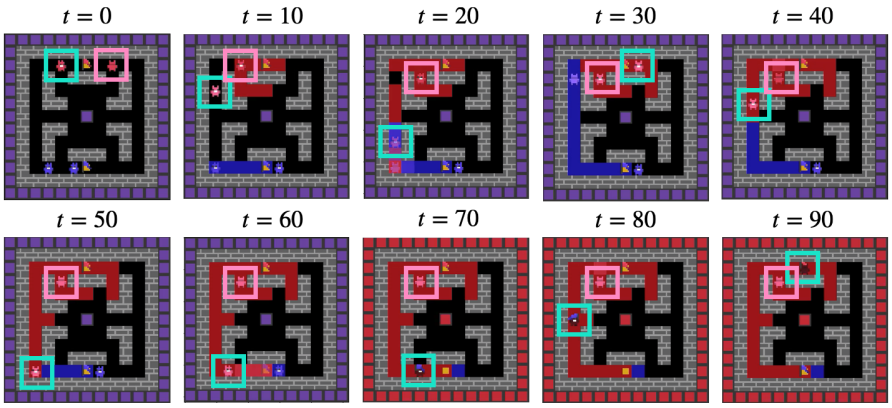


Figure 18: A Capture the Flag trajectory from trained CBP policies. The two red agents have learned emergent roles: attacker (teal square) and defender (pink square).

(rows), each of which is described by a set of features (columns). Lasso neighborhood assumes that each variable can be approximated as a sparse linear combination of the observations (i.e. features) of other variables. For a random variable x_i, this approximation is computed as:

$$\min_{\beta_i} ||\boldsymbol{X}_i - \boldsymbol{X}_{\setminus i}\beta_i||_2^2 + \alpha||\beta_i||_1$$

where $\boldsymbol{X}_i$ represents the features describing the variable x_i (i.e., transpose of the i'th row of X), $\boldsymbol{X}_{\setminus i}$ represents the features from rest of the variables (the remaining rows in $\boldsymbol{X}$, $\boldsymbol{\beta}_i$ is a vector of coefficients, and α weights the L1-regularization term (enforcing sparsity).

Importantly, coefficients $\boldsymbol{\beta} i$ determine which edges, if any, are connected to the node that represents x_i. In particular, for some additional variable x_j, an edge is established between x_i and x_j if either β_{ij} or β_{ji} is non-zero (or both). Intuitively, because a graph is a representation of pairwise relationships, Lasso neighborhood selection posits that learning a graph is equivalent to learning a neighborhood for each vertex—i.e., the other vertices to which it is connected—assumes that the observation at a particular vertex may be represented by observations at the neighboring vertices.

In this work, we construct an observation matrix from the outcomes of interventions and use Lasso neighborhood selection to model the relationships (i.e. similarities and dissimilarities) between interventions.