Concept-based Understanding of Emergent Multi-Agent Behavior

Anonymous Authors[1]

Abstract

This work studies concept-based interpretability in the context of multi-agent reinforcement learning (MARL). We propose Concept Bottleneck Policies (CBPs) as a method for learning intrinsically interpretable, concept-based policies with MARL. We demonstrate that, by enabling each agent's policy to be conditioned on its own estimates of human-understandable concepts from construction, our method enables behavioral analysis via concept intervention. Experiments show that concept intervention over CBPs enables understanding behaviors and inter-agent social dynamics in both cooperative and mixed-motive settings. Our approach can reliably detect successful emergent coordination, coordination failures (lazy agents), and dependencies between agents' behavior. Moreover, we demonstrate that our approach matches 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—e.g., Go (Silver et al., 2016), Chess and Shogi (Silver et al., 2017), Stratego (Perolat et al., 2022))—and online games—e.g., StarCraft (Vinyals et al., 2019), Dota2 (Berner et al., 2019)—multi-agent learning has enabled agents to develop a range of capabilities, such as navigating social dilemmas (Leibo et al., 2017), balancing low-level control and high-level strategy in football (Liu et al., 2022b), and even learning to cooperate with humans (Carroll et al., 2019).

For all its success, multi-agent learning still lacks in a critical area: interpretability. Understanding emergent multi-agent behavior is challenging—for one, it is difficult to decipher the nature of a learned coordination strategy (or whether agents have learned to coordinate at all)—and these issues are only compounded by the inherent opacity of neural networks, which state-of-the-art multi-agent methods rely upon heavily. For this reason, recent work has shifted focus from traditional measures of performance (reward, evaluations against human experts, etc.) to better understanding emergent behaviors (Omidshafiei et al., 2022). Related methods perform behavioral analysis in a *post-hoc* manner, either by visualizing trajectories (Baker et al., 2019), quantifying basic statistics related to agent behavior (Liu et al., 2022b), or measuring the predictability of game-related concepts from a network's activations (McGrath et al., 2021).

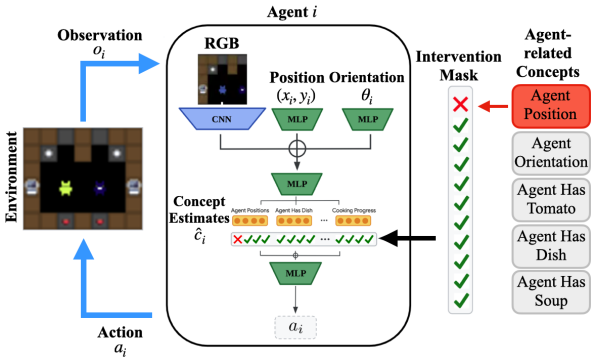


Figure 1. Concept Bottleneck Policies distill agent decisions into human-understandable concepts, yielding interpretable, concept-based policies for MARL and enabling behavioral analysis via concept intervention.

In the supervised learning literature, an alternative to post-hoc analysis is *intrinsic interpretability*, where a model's decisions incorporate human-understandable concepts directly. Such models may offer better alignment with intended concepts than post-hoc methods (Rudin, 2019). Of particular relevance is the work of Koh et al. (2020), which showed that it is possible to train an end-to-end neural network that is "bottlenecked" by human-understandable concepts, enabling intrinsic, concept-based interpretability while maintaining high performance in image classification tasks.

In this work, we bridge the multi-agent reinforcement learning (MARL) and concept-based interpretability domains by introducing Concept Bottleneck Policies (CBPs) as a class of intrinsically-interpretable, concept-based models for MARL. CBPs force agents to make decisions by first

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

predicting an intermediate set of human-understandable concepts, then using those concepts to select actions. This architecture not only makes multi-agent decision-making immediately transparent—agents reveal the environmental and inter-agent factors influencing each of their actions—but it enables another powerful tool for behavioral analysis: intervention over the agents' own concept estimates.

Experimentally, we show that interventions over CBPs uncover a number of key aspects of learned multi-agent behavior. In cooperative environments, concept interventions distinguish agents who have learned to coordinate from those that act independently, detect 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. Moreover, in mixed-incentive environments, we demonstrate that it is possible to identify inter-agent social dynamics by learning a simple graph over intervention outcomes. More concretely, by iteratively intervening over each concept in an agent's CBP and examining its impact on agent behavior (e.g., reward, actions, environmental features), we can learn a sparse graph representing the pairwise relationship of inter-agent intervention effects. Such graphs capture both high-level role-related information (free loader agents vs. public service agents) and low-level spatial information (agents who interact) simultaneously. Finally, we show that CBPs perform comparably to non-concept based policies on difficult coordination tasks; meaning that CBPs achieve interpretability without sacrificing performance.

In sum, our contributions are as follows: (i) We introduce Concept Bottleneck Policies as an interpretable, concept-based learning architecture for MARL; (ii) We show that concept intervention and graph learning over CBPs enables a better understanding of multi-agent behavior (e.g., coordination, inter-agent social dynamics). (iii) We show that CBPs can learn effective policies that 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 (Simonyan et al., 2013; Smilkov et al., 2017; Sundararajan et al., 2017; Springenberg et al., 2014). Interpretability using grounded concepts has also been explored to provide more meaningful, human-understandable explanations of model decisions (Kim et al., 2018; Ghorbani et al., 2019; Bau et al., 2020; Ghandeharioun et al., 2021; Chen et al., 2020; Yeh et al., 2020; Stammer et al., 2021). As described earlier, concept bottleneck models (Koh et al., 2020) 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 (Bastani et al., 2017; Madumal et al., 2020); or analyze agent behaviors from a post-hoc perspective (Yang et al., 2018; Amir & Amir, 2018; Sequeira & Gervasio, 2020; Liu et al., 2022a; Jaderberg et al., 2019; Omidshafiei et al., 2022; McGrath et al., 2021; Forde et al., 2022). Approaches that directly increase transparency include those that combine model compression and decision trees to yield more interpretable agent policies (Bastani et al., 2017), or rely upon causal decision trees with limited depth to explore causality of agent decisions (Madumal et al., 2020). However, these approaches have not yet been extended to MARL settings with complex observations (e.g., images), where function approximators are typically needed.

Post-hoc RL interpretability methods have used traditional saliency-mapping techniques to better attribute decisions to image observation regions (Yang et al., 2018), highlighting states that lead to major differences in agent behaviors (Amir & Amir, 2018), and summarized key agent behaviors using measures such as action uncertainty (Sequeira & Gervasio, 2020). Increasingly, natural language has been used to aid human understanding of agents through instructions (Ahn et al., 2022) or policy explanations (Hayes & Shah, 2017).

Interpretability of MARL agents has received limited attention in prior works, with primary investigations gauging agents' internal representations by making predictions about future outcomes (Liu et al., 2022a), visualizing latent clusterings of agents' neural activation vectors (Jaderberg et al., 2019; Mahajan et al., 2019), and most recently using offline behavioral analysis to learn behavioral spaces over agents (Omidshafiei et al., 2022). Recent works have also conducted analysis of decision-making in two-player games such as Chess (McGrath et al., 2021), and Hex (Forde et al., 2022). In contrast to our work, these approaches focus on post-hoc interpretability, whereas ours makes agent decisions directly transparent, enabling novel forms of downstream analysis (e.g., coordination/lazy agent detection).

3. Background

Markov Games A partially-observable Markov game (Littman, 1994) 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 $\mathcal{T} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \rightarrow \mathcal{S}$, produces new observations for each agent following the observation function $\mathcal{O} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \rightarrow \boldsymbol{\Omega}$, and emits a reward to each agent defined by the reward function $\boldsymbol{\mathcal{R}} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \rightarrow \mathbb{R}^N$. 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 referred to as the joint policy.

Concept Bottleneck Models Concept bottleneck models (CBMs) are a class of intrinsically-interpretable neural network architecture for supervised learning (Koh et al., 2020). CBMs make predictions by first estimating a set of human-understandable concepts, then producing an output based on those concept estimates–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.

4. Concept Bottleneck Policies for MARL

In this section, we introduce Concept Bottleneck Policies (CBPs) as an intrinsically interpretable, concept-based policy learning method for MARL. First, we bridge concept-based interpretability and MARL by extending the Markov game formalism to support concept-based learning. Then, we introduce the CBP architecture and show how it can be learned within any existing policy learning scheme. Finally, we demonstrate our method's potential for interpreting emergent multi-agent behavior, introducing multiple behavioral analysis techniques that use concept intervention to detect successful coordination, expose coordination failures (e.g., lazy agents), and derive inter-agent social dynamics.

4.1. Concept-based Markov Games

To support concept learning, we extend Markov games in two important ways. First, we assume that, in addition to its 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, such as the position of agents or objects; this assumption holds in popular RL domains like Atari (Mnih et al., 2013) and MeltingPot (Leibo et al., 2021). Second, we allow agents to generate estimates $\hat{\boldsymbol{c}}$ of this concept state as part of their internal representation. In sum, at each time-step t, the environment is described by both its state s_t and concept-state $\boldsymbol{c}_t$, and each agent i produces both an action $a_{i,t}$ and a set of concept estimates $\hat{\boldsymbol{c}}_{i,t}$.

4.2. Concept Bottleneck Architecture

There are many ways in which an agent's concept estimates $\hat{\boldsymbol{c}}_i$ can be modeled within the RL framework. Inspired by CBMs, we examine policy architectures in which an agent's action is conditioned entirely on its own concept estimates (see Figure 1). 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 $a_{i,t} \sim \pi_i^{\text{act}}$. Structuring the network such that actions are conditioned on concepts creates an intrinsically interpretable policy—the policy is forced to provide a human-understandable rendering of the factors driving its own decision making.

Importantly, $\hat{\boldsymbol{c}}_{\boldsymbol{i}}$ can be 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.

4.3. Concept Bottleneck Learning

Under the concept bottleneck 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 the following concept loss:

$$L_C(\boldsymbol{c}, \hat{\boldsymbol{c}}) = \sum_{j=1}^{|C|} L_{C_j}(c_j, \hat{c}_j) \tag{1}$$

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.

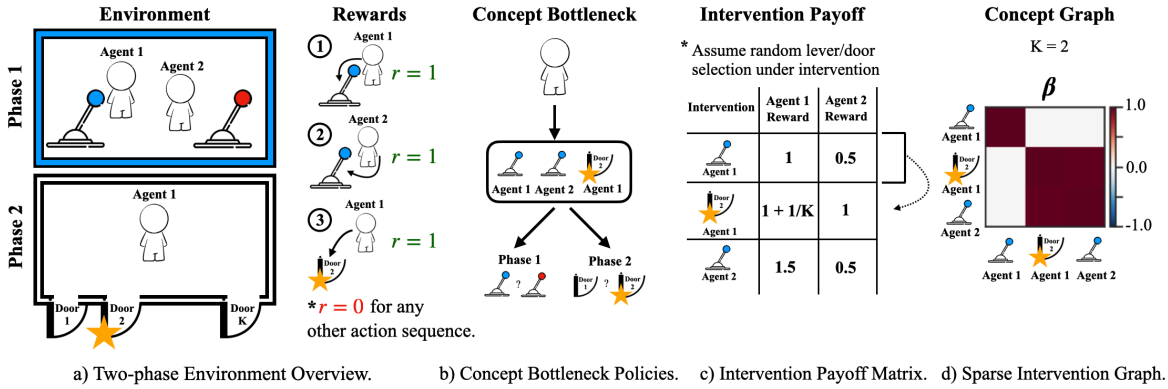
To learn a policy from concept estimates, we attach Equation (1) as an auxiliary loss to the reward-based loss defined by any base RL algorithm (e.g., PPO (Schulman et al., 2017), etc). In general, if L_{RL} is a generic reward-based loss, we construct a joint concept bottleneck policy loss:

$$L_{CBP} = L_{\text{RL}} + \lambda L_C \,, \tag{2}$$

where the concept loss coefficient λ weights the relative importance of concept prediction.

4.4. Behavioral Analysis via Concept Intervention

In addition to its intrinsic interpretability, a key feature of our method is its support of intervention analysis. Once a CBP is trained, we can freeze the policy network, roll out a trajectory, and at each time-step replace an individual concept estimate (e.g., the j-th concept estimate $\hat{c}_j$) with a



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

Figure 2. A toy example of graph learning enabled by Concept Bottleneck Policies. **a)** A two-phase game. In phase one, Agent 1 and 2 choose from two levers (red, blue) using a special observation (wall color). Agents receive individual rewards for selecting the correct lever (blue) in the correct sequence (Agent 1 moves first, then Agent 2). 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)** Learning a graph over these payoffs reveals relationships between concept interventions.

replacement value $\bar{c}_j$. In the simplest case, we replace the estimate completely by setting $\bar{c}_j = 0$. We can then observe the effect, if any, that $\bar{c}_j$ has on the agent's behavior[1].

Detecting Coordination Using concept interventions, we propose a direct test of coordination. Consider two agents: i and j. For i to coordinate with j, i must condition its policy on information about j (j's position, orientation, etc). Thus, if the agents are coordinating and we remove i's concept estimates pertaining to j, we should observe a decrease in their performance. Conversely, if performance 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 Failures CBPs also provide a means for detecting coordination failures, such as 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 examining the extent to which team performance degrades as a result. If performance remains the same with the agent incapacitated, we conclude that it is a lazy agent.

4.5. Modeling Inter-Agent Social Dynamics

In complex, mixed-incentive settings, 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. These interactions are difficult to interpret from rewards alone. Here we propose a simple way to build a sparse graph over CBP intervention outcomes that may aid interpretation.

Let $\text{Int}(\hat{c}_j, i)$ be an intervention over the j'th concept estimate $\hat{c}_j$ of an agent i. Now, suppose we take each intervention to be a random variable that is described by some agent-related features, such as the average reward $\boldsymbol{r}_{\hat{\boldsymbol{c}}_j} = \{r^1_{\hat{c}_j}, ..., r^n_{\hat{c}_j}\}$ each agent receives under the intervention[2]. By performing $\text{Int}(\cdot)$ iteratively over each concept $c \in C$ and for each agent $i \in \{1, ..., N\}$ ($N \times |C|$ agent-concept pairs in total), it is possible to construct a matrix $\boldsymbol{X}$ wherein each row is the outcome of an intervention:

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

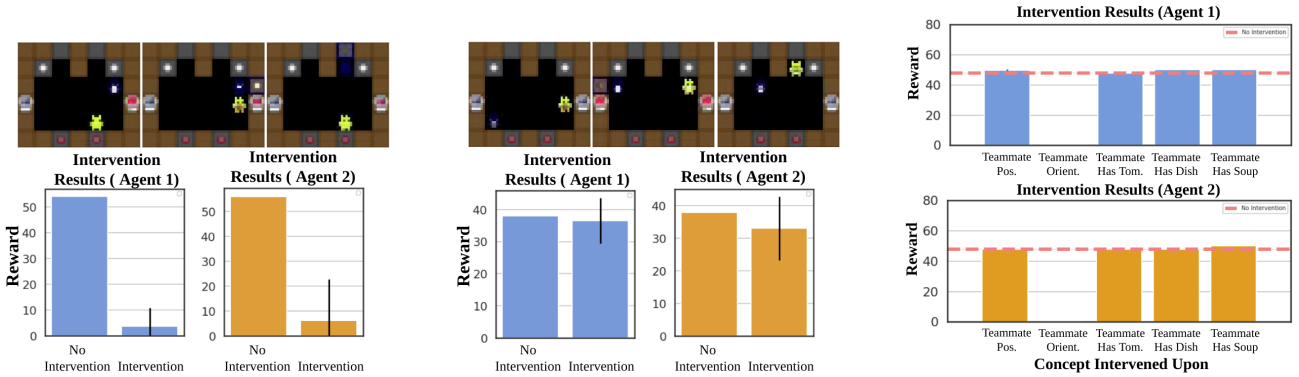
We can then learn a graph that encodes the pairwise relationships between interventions by performing Lasso neighborhood selection (Meinshausen & Bühlmann, 2006) over $\boldsymbol{X}$. Specifically, for the intervention $\text{Int}(\hat{c}_j, i)$, we solve:

$$\min_{\beta_{ij}} ||\boldsymbol{X}_{ij} - \boldsymbol{X}_{\setminus ij}\beta_{ij}||_2^2 + \alpha||\beta_{ij}||_1 \qquad (3)$$

where $\boldsymbol{X}_{ij}$ is the outcome of $\text{Int}(\hat{c}_j, i)$, $\boldsymbol{X}_{\setminus ij}$ represents the remaining $N \times |C| - 1$ interventions, and α weights L_1-regularization, 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 $\boldsymbol{\beta}_{ij}$ establish the similar-

[1] We carefully examine whether this leads to OOD examples and propose ways to address this (see Appendix D.3).

[2] Intervention outcomes can be defined in terms of other features, such as resources collected, proximity to other agents, etc.



(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. Moreover, for the policies shown in (a), intervening over each agent's teammate-related concepts individually (c) identifies the inter-agent factors driving the team's coordination strategy. Surprisingly, teammate orientation is relied upon heavily by both agents for coordination.

ity/dissimilarity of Int$(\hat{c}_j, i)$'s outcome to other interventions. These relationships may hint at the nature of agent behavior. We provide a comprehensive overview of Lasso neighborhood selection in Appendix E.

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 designated as a reward-giving lever (blue in this example). An observable feature (wall color) indicates which is the reward-giving lever. Agent 1 acts first and must choose which lever to pull, receiving an individual reward ($r = 1.0$) for selecting the correct lever. Agent 2 acts second, 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 the door from which it will exit the room. Exiting from the 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.

Both agents are initialized with a CBP (see Figure 2b)). In Agent 1's CBP, actions are conditioned on two concepts: an estimate of lever color, and and estimate of the door indicator. Agent 2's CBP estimates lever color alone. We assume that intervening over any concept reduces decision-making to a random policy. Using this information, we can compute the expected rewards for each agent under intervention, as shown in Figure 2c. Computing Equation (3) 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 information 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 Concept Bottleneck Policies can answer 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 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?

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 (leading to poor generalization)?

Bottleneck Performance Do CBPs perform as well as traditional, non-concept-based policies; thereby achieving intrinsic interpretability without sacrificing performance?

Setup: We use two environments from Melting Pot (Leibo et al., 2021) for our experiments. Melting Pot offers a suite of multi-agent environments that vary along a number of axes, including: incentive alignment (cooperative, mixed motive, adversarial), coordination requirements, etc. It therefore provides a strong benchmark for studying the many facets of emergent behavior. Specific cooperative and mixed-incentive environments are described in the subsections below, and further details, including the complete set of concepts for each environment, are provided in Appendix B. For learning, we use PPO (Schulman et al., 2017)

as a base algorithm, which has been shown to be state-of-the-art for multi-agent settings (de Witt et al., 2020; Yu et al., 2021), and train in a fully-decentralized fashion. We augment the PPO objective with the concept loss defined in Equation (1) (we refer to this combination as ConceptPPO moving forward). Additional training details and hyperparameter sweeps are provided in Appendix C.

5.1. Cooperative: Identifying Emergent Coordination

We evaluate CBPs as a tool for understanding emergent multi-agent 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? First, we aim to distinguish policies that learn coordination from those that learn to solve the task independently. We evaluate each of the trained ConceptPPO policies over 100 test-time trajectories (and five seeds each) while intervening over all of the concepts related to each agent's teammate. We measure the average cumulative reward attained by the agents under intervention. Results of this analysis are shown in Figure 3.

The cooking environment in Figure 3 has an open floor plan with separate sets of ingredients (tomatoes) and tools (bowls, pots). 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 through role assignment. In particular, the orange agent works the bottom-right corner of the environment picking up tomatoes and bringing them to the cooking pot; while the blue agent stays in the top-right corner running dishes to and from the pot to deliver soup. Intervention over teammate-related concepts leads to a catastrophic drop in performance, as an 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 performance, 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 D.2 we show that, by averaging across all policies trained in each Melting Pot cooking environment, concept intervention can also be used to identify the extent to which a particular environment requires coordination.

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. Rather than intervening over all teammate-related concepts at once, we intervene one at a time. If agents are coordinating using this signal, the intervention should result in a sharp drop in performance. Results are shown in Figure 3c.

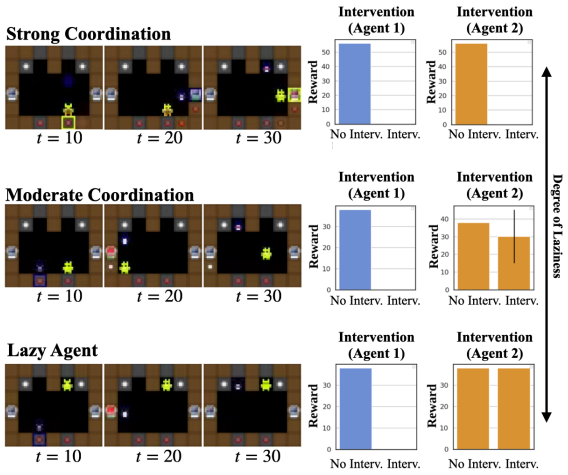


Figure 4. Intervening on an agent's concepts about itself exposes its *degree of laziness*. With strong coordination, intervention has a large negative impact on the performance of each agent. When an agent is not productive, intervention impact is negligible.

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—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 by using an empirical sample of the 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 D.3.1). Second, we train additional ConceptPPO policies without orientation as a concept and re-run this iterative intervention. These results show that, without orientation, agent coordination latches onto another concept (teammate has_soup) as its primary signal (see Appendix D.3.2). We emphasize that identifying the specific factors that underlie agent coordination would be more difficult without the concept intervention enabled by CBPs.

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.

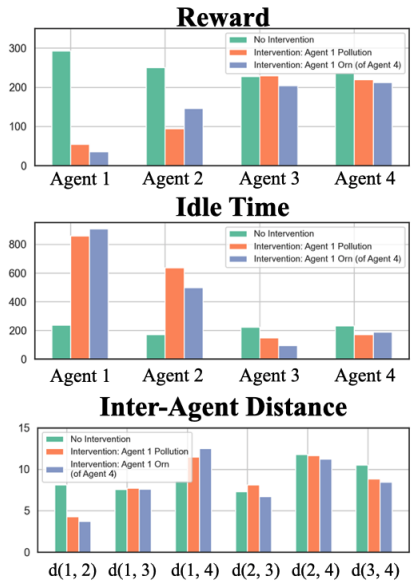


Figure 5. Statistics collected over Clean Up trajectories (with and without intervention). Under intervention, a relationship between Agent 1 and Agent 2's reward, idle time, and inter-agent distance is exposed, indicating a collision between the agents. Raw statistics alone, therefore, suggests that Agents 1 and 2 are coordinating.

The magnitude of performance degradation is a direct function of the productivity of each agent. When agents are coordinating (top row), concept intervention has a strong negative impact on the performance of both agents. In the extreme lazy agent case in which one agent does not move over the course of a trajectory (bottom row), reward for that agent 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 row). Thus, these results not only demonstrate that the proposed concept bottleneck architecture has given us a unique way of diagnosing lazy agents, but also suggests that it can quantify the *degree of laziness* of each agent a function of performance degradation.

5.2. Social Dilemmas: Inter-agent Social Dynamics

We further evaluate CBPs by studying the extent to which they discover inter-agent social dynamics in mixed-motive environments. As a representative example, we use MeltingPot's Clean Up environment, a social dilemma in which agents must balance selfish behavior (harvesting fruit) with public service (cleaning a river is necessary for fruit to grow). This game therefore extends the scope of emergent social interactions to exploitation, free-riding, and public resource sharing; which is a good benchmark for our method.

A natural way to investigate inter-agent dynamics with CPBs is by looking at raw behavioral statistics (Figure 5) with and without intervention. As seen in Collaborative Cooking, this alone can reveal interesting insights. For example, Figure 5

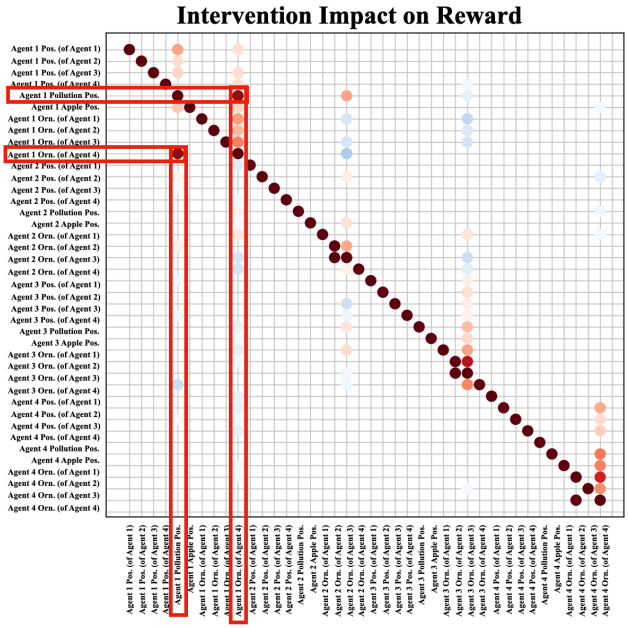


Figure 6. A sparse graph learned over concept interventions (using average reward as features). A strong bi-directional relationship between two concepts: Agent 1's closest pollution, and Agent 1's estimate of Agent 4's orientation.

suggests that Agent 1 and 2 are dependent on each other in some way, as their reward and idle time appears correlated via an intervention. Moreover, we see that both the decrease in Agent 1 and 2's reward and the increase in their idleness is further related to a decrease in their average inter-agent distance, indicating that this interaction likely involves spatial coordination. A reasonable conclusion, therefore, is that Agent 1 and Agent 2 require accurate estimations of each other (e.g., position, orientation) to complete the task.

However, we posit that raw statistics alone do not tell the whole story and, in fact, may obscure details about the *chains of social interactions* that lead to these intervention outcomes. To investigate, we build a series of graphs by computing Equation (3) over all agent-concept interventions using average reward, pairwise inter-agent distance, and the number of idle (non-moving) steps, respectively. Figure 6 shows the graph computed over average rewards (in matrix form), which exposes a number of intriguing concept-to-concept relationships that exist across agents.

First, there are *no* identifiable relationships between Agent 1 and 2 (as evidenced by the lack of edges linking 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. Coincidentally, this same prominent bi-directional relationship also exists in graphs computed over inter-agent

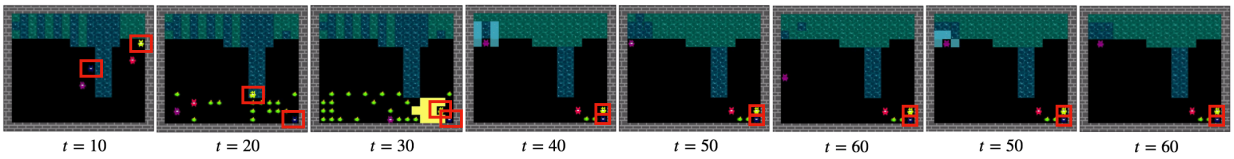


Figure 7. A qualitative example of behavioral change (collision) under intervention.

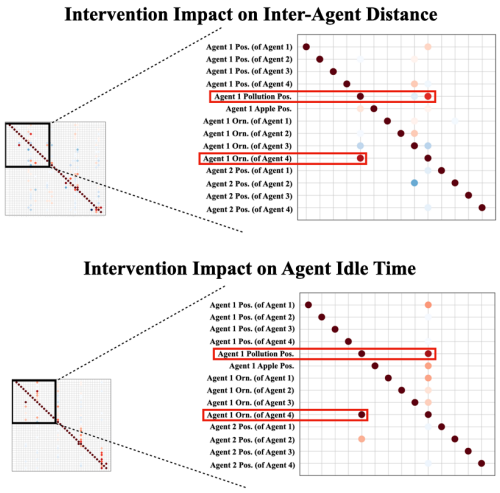


Figure 8. Sparse graphs over intervention outcomes for inter-agent distance and agent idleness. The upper quadrant of each graph shows the same bi-directional relationship as in the reward graph.

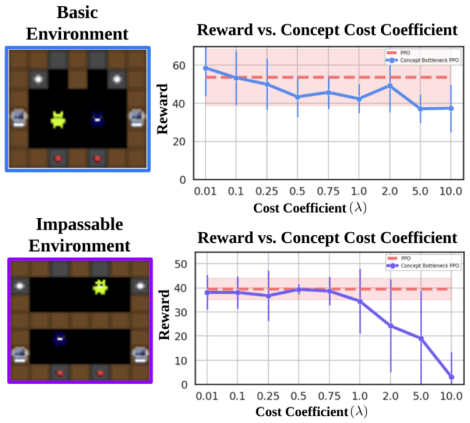


Figure 9. Asymptotic reward of ConceptPPO vs. PPO. For $\lambda \leq 0.5$, ConceptPPO matches the performance of non-concept-based PPO.

distance and agent idleness features (see Figure 8).

The graphs reveal much deeper inter-agent relationships, exposing a *chain of events that links the emergent behaviors of each agent*. Rather than Agent 1 and Agent 2 coordinating, as the raw statistics suggest, the graph shows that their relationship is in fact the result of an incidental interaction in the environment. Concretely, intervening over both Agent 1's pollution concept and Agent 1's orientation estimate of Agent 4 lead to the same outcome: a collision between Agent 1 and Agent 2. This is in turn why we see a correlation between their rewards under intervention. This reveals that Agent 1 and Agent 2 have a brittle coordination strategy that is easily disrupted; and is confirmed by inter-agent distance in raw behavioral statistics (Figure 5) and in visual inspection of the game (Figure 7).

In sum, our graph learning strategy over CBP interventions reveals complex inter-agent dynamics that could have taken humans many hours to detect. An important future work is to extend this simple technique to more complex graph learning paradigms(Dong et al., 2019).

5.3. 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 9 provides an overview of these results. We find 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 9 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 Equation (1) ($\nabla L_{RL} + \lambda \nabla L_C$) 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.

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. CBPs achieve comparable levels of performance to traditional non-concept-based policy learning methods.

References

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Amir, D. and Amir, O. Highlights: Summarizing agent behavior to people. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1168–1176, 2018.

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., and Mordatch, I. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

Bastani, O., Kim, C., and Bastani, H. Interpretability via model extraction. *arXiv preprint arXiv:1706.09773*, 2017.

Bau, D., Zhu, J.-Y., Strobelt, H., Lapedriza, A., Zhou, B., and Torralba, A. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–30078, 2020.

Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Carroll, M., Shah, R., Ho, M. K., Griffiths, T., Seshia, S., Abbeel, P., and Dragan, A. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

Chen, Z., Bei, Y., and Rudin, C. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, 2020.

de Witt, C. S., Gupta, T., Makoviichuk, D., Makoviychuk, V., Torr, P. H., Sun, M., and Whiteson, S. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Dong, X., Thanou, D., Rabbat, M., and Frossard, P. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

Forde, J. Z., Lovering, C., Konidaris, G., Pavlick, E., and Littman, M. L. Where, when & which concepts does alphazero learn? lessons from the game of hex. In *AAAI Workshop on Reinforcement Learning in Games*, volume 2, 2022.

Games, G. T. Overcooked. https://store.steampowered.com/app/ 448510/Overcooked/, 2016.

Ghandeharioun, A., Kim, B., Li, C.-L., Jou, B., Eoff, B., and Picard, R. W. Dissect: Disentangled simultaneous explanations via concept traversals. *arXiv preprint arXiv:2105.15164*, 2021.

Ghorbani, A., Wexler, J., Zou, J. Y., and Kim, B. Towards automatic concept-based explanations. *Advances in Neural Information Processing Systems*, 32, 2019.

Hayes, B. and Shah, J. A. Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pp. 303–312. IEEE, 2017.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castaneda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pp. 2668–2677. PMLR, 2018.

Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., and Liang, P. Concept bottleneck models. In *International Conference on Machine Learning*, pp. 5338–5348. PMLR, 2020.

Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037*, 2017.

Leibo, J. Z., Dueñez-Guzman, E. A., Vezhnevets, A., Agapiou, J. P., Sunehag, P., Koster, R., Matyas, J., Beattie, C., Mordatch, I., and Graepel, T. Scalable evaluation of multi-agent reinforcement learning with melting pot. In *International Conference on Machine Learning*, pp. 6187–6199. PMLR, 2021.

Littman, M. L. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pp. 157–163. Elsevier, 1994.

Liu, S., Lever, G., Wang, Z., Merel, J., Eslami, S. A., Hennes, D., Czarnecki, W. M., Tassa, Y., Omidshafiei, S., Abdolmaleki, A., et al. From motor control to team play in simulated humanoid football. *Science Robotics*, 7(69): eabo0235, 2022a.

Liu, S., Lever, G., Wang, Z., Merel, J., Eslami, S. M. A., Hennes, D., Czarnecki, W. M., Tassa, Y., Omidshafiei, S., Abdolmaleki, A., Siegel, N. Y., Hasenclever, L., Marris, L., Tunyasuvunakool, S., Song, H. F., Wulfmeier, M.,

Muller, P., Haarnoja, T., Tracey, B., Tuyls, K., Graepel, T., and Heess, N. From motor control to team play in simulated humanoid football. *Science Robotics*, 7(69): eabo0235, 2022b. doi: 10.1126/scirobotics.abo0235.

Madumal, P., Miller, T., Sonenberg, L., and Vetere, F. Explainable reinforcement learning through a causal lens. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 2493–2500, 2020.

Mahajan, A., Rashid, T., Samvelyan, M., and Whiteson, S. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems*, 32, 2019.

McGrath, T., Kapishnikov, A., Tomašev, N., Pearce, A., Hassabis, D., Kim, B., Paquet, U., and Kramnik, V. Acquisition of chess knowledge in alphazero. *arXiv preprint arXiv:2111.09259*, 2021.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Omidshafiei, S., Kapishnikov, A., Assogba, Y., Dixon, L., and Kim, B. Beyond rewards: a hierarchical perspective on offline multiagent behavioral analysis. *arXiv preprint arXiv:2206.09046*, 2022.

Perolat, J., de Vylder, B., Hennes, D., Tarassov, E., Strub, F., de Boer, V., Muller, P., Connor, J. T., Burch, N., Anthony, T., et al. Mastering the game of stratego with model-free multiagent reinforcement learning. *arXiv preprint arXiv:2206.15378*, 2022.

Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sequeira, P. and Gervasio, M. Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. *Artificial Intelligence*, 288:103367, 2020.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Stammer, W., Schramowski, P., and Kersting, K. 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*, pp. 3619–3629, 2021.

Strouse, D., McKee, K., Botvinick, M., Hughes, E., and Everett, R. Collaborating with humans without human data. *Advances in Neural Information Processing Systems*, 34: 14502–14515, 2021.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

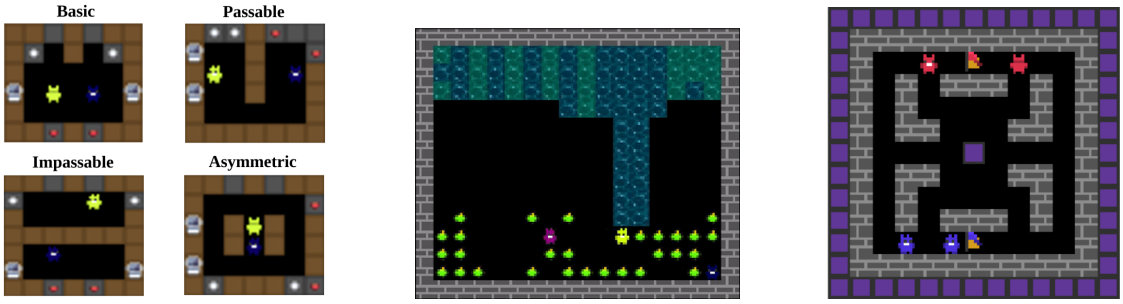
Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Wu, S. A., Wang, R. E., Evans, J. A., Tenenbaum, J. B., Parkes, D. C., and Kleiman-Weiner, M. Too many cooks: Bayesian inference for coordinating multi-agent collaboration. *Topics in Cognitive Science*, 13(2):414–432, 2021.

Yang, Z., Bai, S., Zhang, L., and Torr, P. H. Learn to interpret atari agents. *arXiv preprint arXiv:1812.11276*, 2018.

Yeh, C.-K., Kim, B., Arik, S., Li, C.-L., Pfister, T., and Ravikumar, P. On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems*, 33:20554–20565, 2020.

Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., and Wu, Y. 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 10. Environment Overview.

A. Discussion

A.1. 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.

A.2. Societal Implications

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.

A.3. 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).

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 10a. Based on the game Overcooked (Games, 2016) and subsequent work that has developed Overcooked-like environments for studying multi-agent coordination (Carroll et al., 2019; Wu et al., 2021), 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 (Wu et al., 2021; Carroll et al., 2019; Strouse et al., 2021) 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 (Leibo et al., 2017). 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 10b. 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 10c. Capture the Flag is a well-known competitive environment in the multi-agent literature (Jaderberg et al., 2019). 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); (iv) whether or not an agent has the opponent's flag (binary) (scalar); (v) floor paint color (categorical); (vi) flag indicator tile color (categorical).

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	$\{0.75, 0.9, \mathbf{1.0}\}$

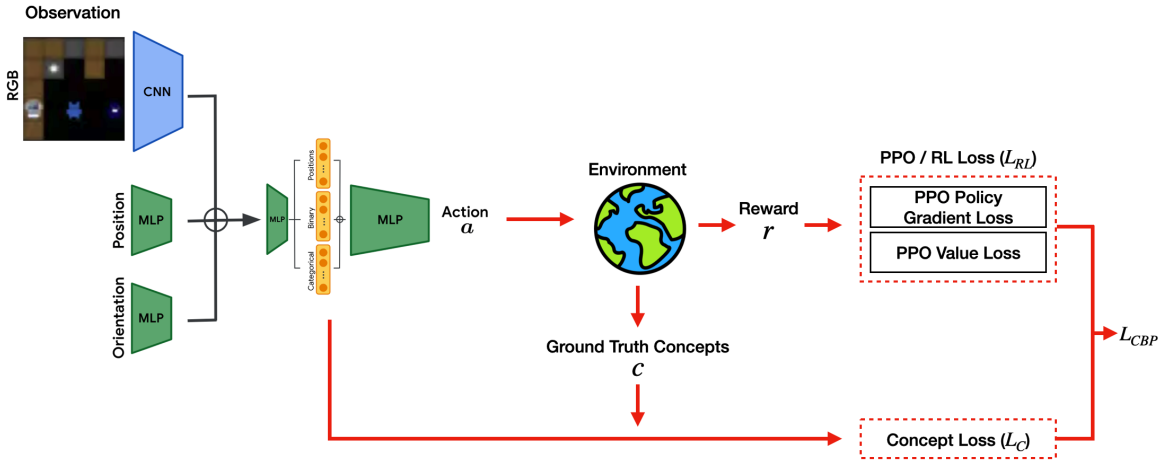


Figure 11. An architecture diagram showing how the concept bottleneck loss L_{CBP} is computed in Equation (1). Crucially, PPO is applied only over L_{RL}, which captures environmental rewards (e.g., delivering in collaborative cooking). Separately, concept loss L_C is computed in a supervised fashion 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.

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, Equation (1). 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.

Figure 12. Concept loss during training in Collaborative Cooking. We plot the average L_C during training for two values of the concept cost coefficient: $\lambda = 0.1$, $\lambda = 0.25$. These policies are the same ones that were used for the intervention analysis in Section 5.1.

We are leveraging PPO to optimize each agent's 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.

D. Additional Experiments

D.1. Concept Loss

Here we examine how concept loss L_C, as measured by Equation (1), is optimized during training. We focus concretely on two pools of policies, those trained with concept cost coefficient $\lambda = 0.1$ and those trained with $\lambda = 0.25$, which correspond to the set of policies that were used for the intervention analyses presented in Section 5.1. Concept loss is averaged across all policies trained with these λ values. Figure 12 shows the result of this analysis. Overall, we find the concept loss does decrease over time, eventually converging after approximately 2000 training steps. The impact of λ is visible through the magnitude of L_C early on in training—concept loss starts at a lower level for $\lambda = 0.25$ and converges to a lower asymptotic value. Though we cannot entirely rule out concept leakage through this analysis alone, this gives us confidence that agents are indeed learning to predict concepts accurately.

D.2. Environmental Demands of Coordination

Environmental Coordination Demands 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 13. 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.

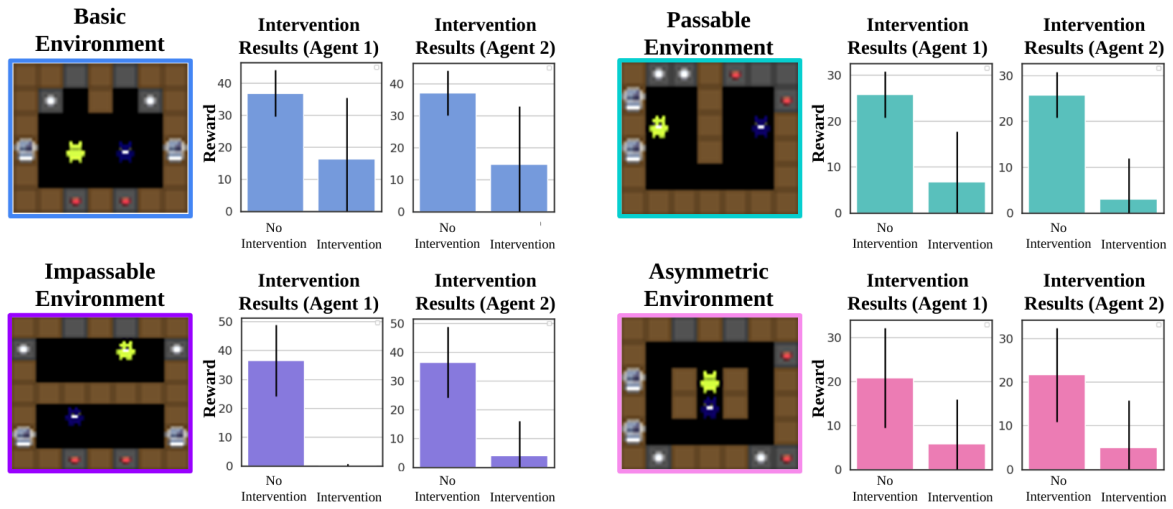


Figure 13. **Does the environment require coordination?** Averaging the impact of concept intervention over all policies trained in an environment reveals the extent to which coordination is required by that environment. In the impassable environment, agents cannot solve the task without coordinating, 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.

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.

D.3. Factors of Coordination Analysis (cont'd)

D.3.1. 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 14, 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 14, 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

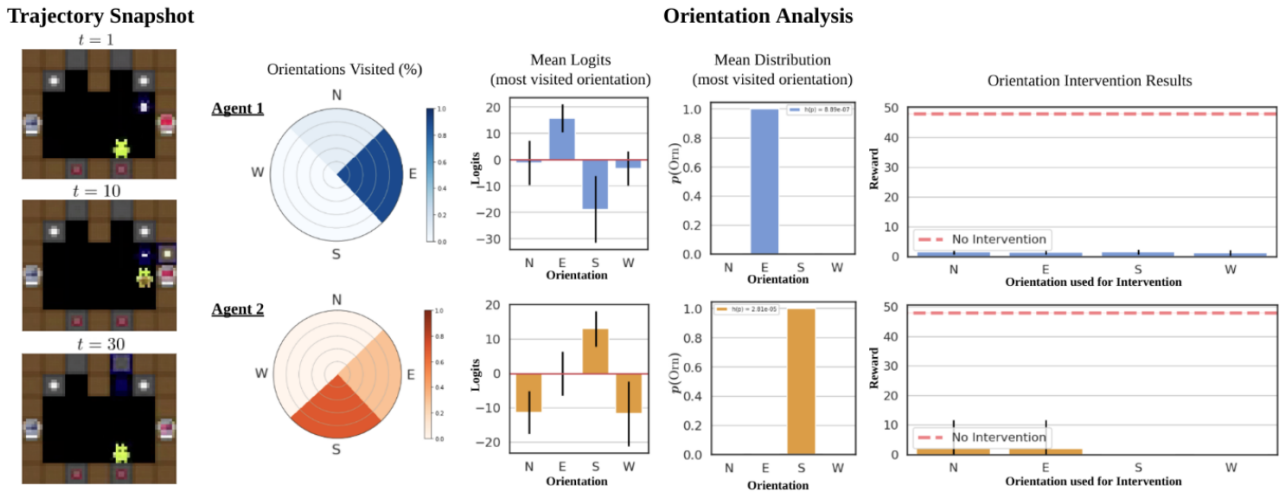


Figure 14. Overview of our method for constructing in-distribution concept masks for intervention. Using test-time trajectories, we compute an empirical distribution of the orientations experienced by each agent. Next, we compute the mean logits (and corresponding distribution) produced by each agent 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.

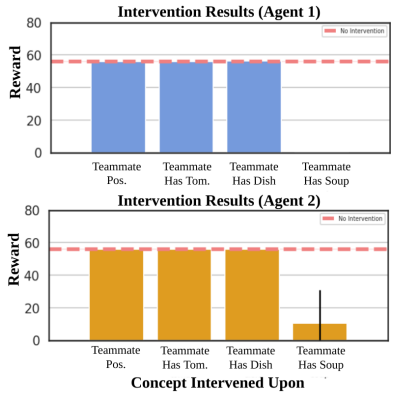


Figure 15. Results of iterative concept intervention on Concept PPO agents trained without orientation.

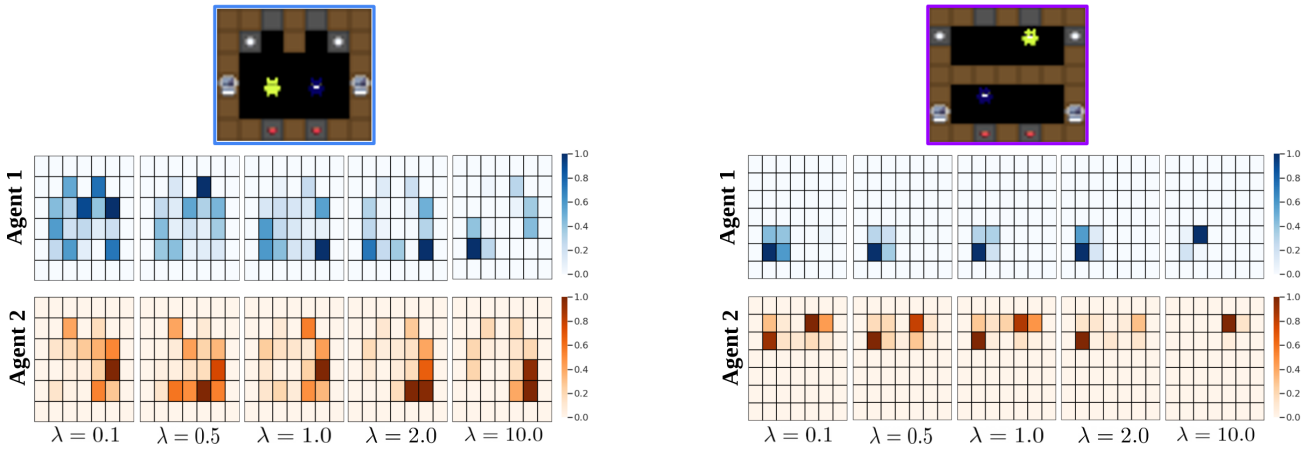
adversarial or OOD example.

D.3.2. 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 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 15. 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.

D.4. 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



(a) Basic Environment. (b) Impassable Environment.

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

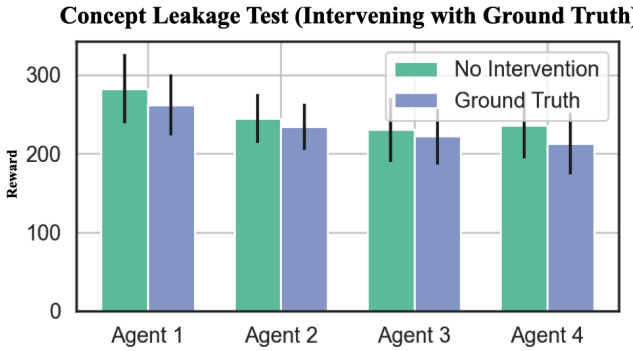


Figure 17. Concept leakage test. We perform an intervention over each agent's concept estimates with the ground truth concept labels from the environment. There is a very slight decrease in performance during intervention, which indicates that some concept leakage does exist. However, the small amount of performance degradation indicates that agents are faithfully learning to estimate concepts.

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.

D.5. Concept Leakage Analysis: Ground Truth Interventions

To examine the extent to which concept leakage occurs as a result of the joint objective outlined in Equation (1), 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 5.2.

The results of this analysis are shown in Figure 17, 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.

D.6. Emergence of Strategic Behavior

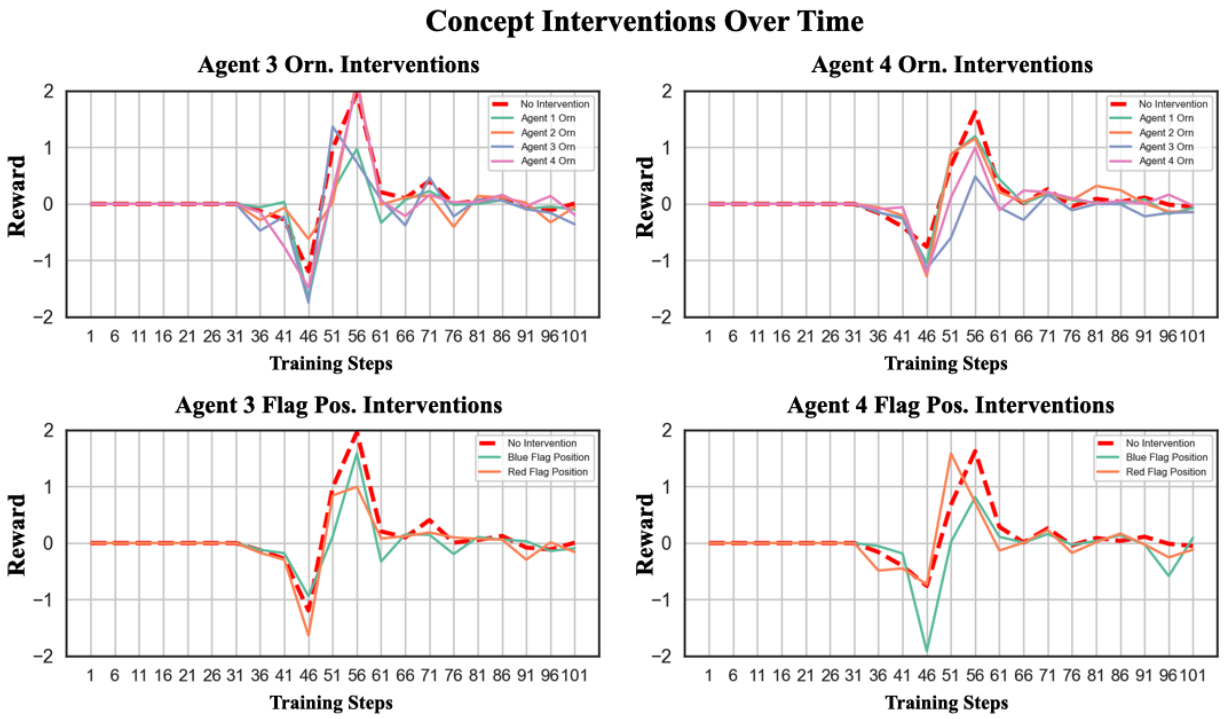


Figure 18. Exposing the emergence of strategic behavior through concept intervention. Concept interventions are performed intermittently over time for each agent. Results for the agents on the red team (Agent 3, Agent 4) are shown here. In the top row, we show results of an intervention over agent-related concepts (orientation), and in the bottom row we show results of an intervention over flag-related concepts (flag position). Honing in on checkpoint 56, where the red agents first receive positive reward, we find an interesting pattern in the intervention results. 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 therefore interacts with each agent and each flag) and Agent 3 is likely a defending agent (and interacts only with its own flag and an attacking blue agent).

Recent work has shown the training MARL agents in competitive environments induces an automatic skill learning curriculum (Baker et al., 2019) similar to that of self-play style training in board games (Silver et al., 2017). Though this multi-agent variant of self-play can lead to the emergence of highly-sophisticated behaviors, such as tool use and offensive/defensive strategy (Baker et al., 2019), it is difficult to decipher when during training specific skills emerge.

Motivated by this powerful yet difficult to interpret setting, we investigate the extent to which CBPs, and specifically intervention over CBPs during training, can expose strategic behaviors as they emerge in multi-agent self-play. To study this, we leverage the Capture the Flag environment outlined in Appendix B.4. We initialize each agent with a separate CBP at the beginning of training and intermittently perform concept interventions over each agent (intervening over each concept estimate from an agent one at a time). As before, we measure the impact of each intervention as the magnitude of reward degradation each agent experiences under intervention.

Results from this analysis are shown in Figure 18, where 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 18, 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 46, 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 56, as it is the first time we see the red team learn productive behavior. At checkpoint 56, 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

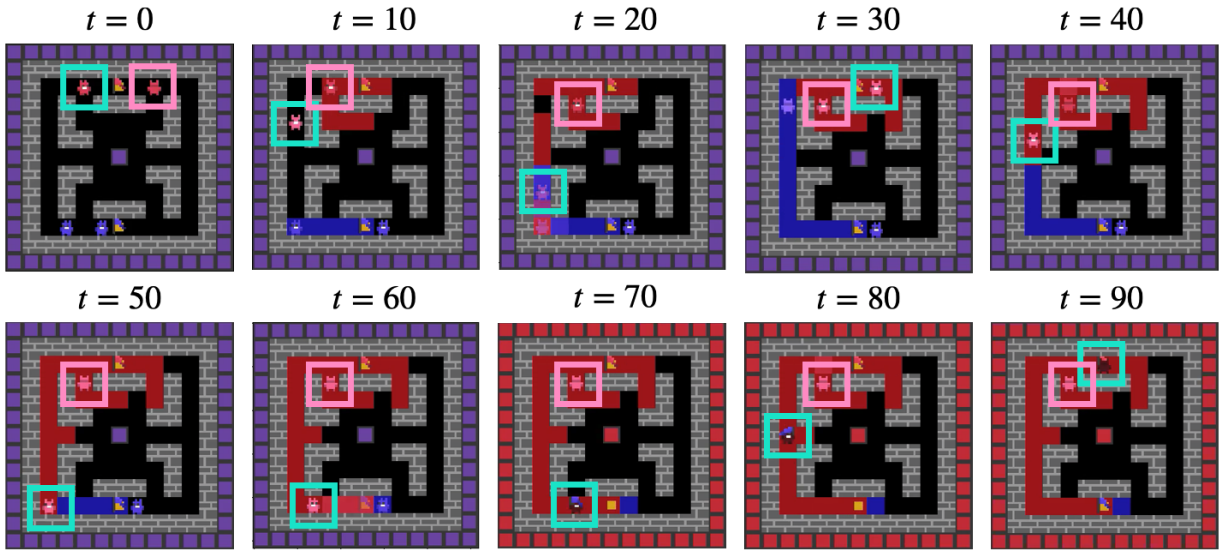


Figure 19. A Capture the Flag trajectory from the trained policies at checkpoint 56. Agent 3 is highlighted in pink and Agent 4 in light-blue. Here, Agent 3 and Agent 4 follow distinct roles: Agent 3 base camps and defends the red flag against a blue attacker; while Agent 4 attacks the blue camp, eventually securing the blue flag and bringing it back to base.

top row of Figure 18, 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 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 18. 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.

Figure 19 provides a snapshot of the agents' behavior (where Agent 4 is highlighted in light-blue, and Agent 3 in pink). As predicted by our intervention analysis, Agent 4 takes on an attacking role—traversing the arena down to the blue team's base (fighting blue agents along the way), capturing the flag, and returning it to the red team's base. Agent 3, on the other hand, takes on a defending role: camping at the red team's base and defending it from a blue attacker—at $t = 30$ you can see Agent 3 prepared to defend against the attacking blue agent, and at $t = 40$ the blue agent has been defeated, clearing the path for Agent 4 to attack.

These results therefore confirm that, in addition to the benefits of post-hoc intervention analysis (from our previous experiments), intervention can be used *during training* to investigate role assignments and strategic behaviors *as they emerge*. These insights are significantly harder to identify from reward curves or exhaustive visualization alone.

D.7. Comparison to Sequential Bottleneck Architecture

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. (2020) compares three architectures for concept bottlenecks in supervised learning settings that lean these functions in different ways:

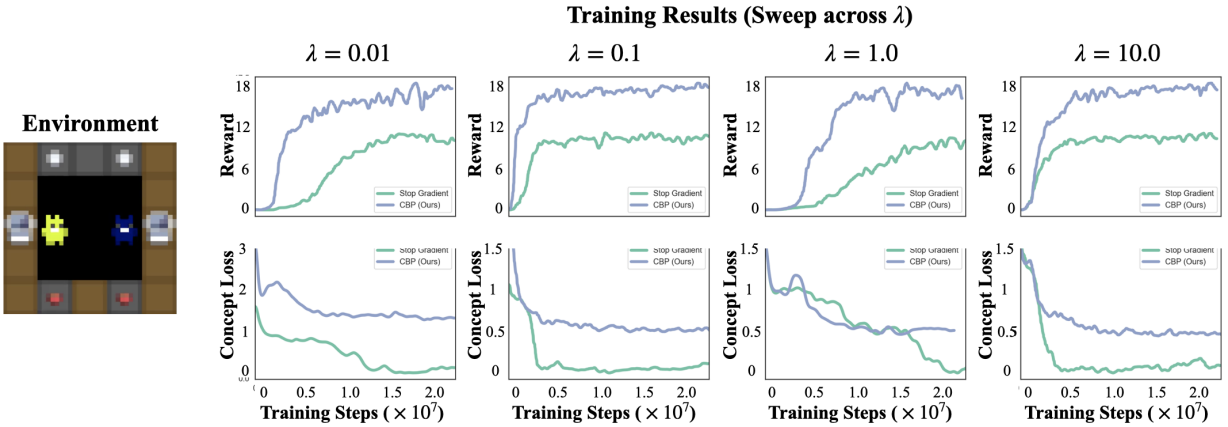


Figure 20. A comparison of CBPs (our proposed joint bottleneck architecture) to the sequential bottleneck architecture across a variety of λ values in a simplified version of the cooking basic environment.

- **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 11) and objective (defined by Equation (1)) 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 cooking environment shown in Figure 20. Comparing performance (in terms of both reward and concept loss) across a range of λ values reveals an interesting pattern: CBPs (our architecture) consistently outperform the sequential bottleneck (implemented via stop gradient). Note that, because this cooking environment is a simplified version of the Cooking Basic environment, λ has a negligible impact on performance for CBPs (as compared to the more difficult environments in Figure 9). Digging in further (see Figure 21, we find that each of the sequential bottlenecks converge to a local minima where only one agent learns to solve the task at a time. This is also why the asymptotic reward for each of the the sequential bottleneck runs in Figure 20 conspicuously lies at about half of reward of the CBP runs.

This pattern of results can be explained as follows: First, as a result of the stop gradient operation in the concept estimator, the sequential bottleneck is technically solving an easier concept estimation problem; thereby converging to a low concept loss more quickly. This stabilizes the network's concept predictions, which are the inputs to the policy network head. In turn, the policy head does not explore as much, and so is more susceptible to getting stuck in a local optima. This local optima is difficult to tune around in practice, resulting in the sub-optimal behavior for the sequential bottleneck shown in both Figure 20 and Figure 21.

E. Lasso Neighborhood Selection

Lasso neighborhood selection is a simple method that poses graph learning as a Lasso regression problem (Dong et al., 2019). Let $\boldsymbol{X}$ be an observation matrix that is composed of some set of random variables (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

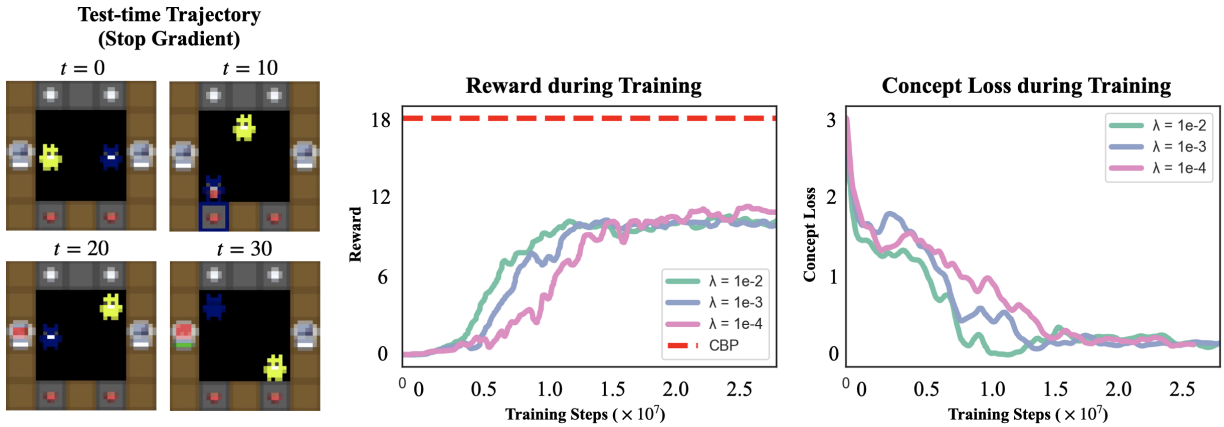


Figure 21. **Left**: An example trajectory from the sequential bottleneck architecture. Notably, only one agent ever learns to solve the task at a time. **Right**: Results for the sequential bottleneck for additional λ values.

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.