Concept Learning for Interpretable Multi-Agent Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract: Multi-agent robotic systems are increasingly operating in real-world environments in close proximity to humans, yet are largely controlled by policy models with inscrutable deep neural network representations. We introduce a method for incorporating interpretable concepts from a domain expert into models trained through multi-agent reinforcement learning, by requiring the model to first predict such concepts then utilize them for decision making. This allows a human operator to both query the resulting concept policy models to obtain a decision making rationale, as well as intervene in the event concepts are predicted incorrectly. We show that this yields improved interpretability and training stability, with benefits to policy performance and sample efficiency in a simulated and real-world cooperative-competitive multi-agent game.

Keywords: Multi-Agent Reinforcement Learning, Interpretable Machine Learning

1 Introduction

With burgeoning adoption in fields such as autonomous driving, service robotics, and healthcare, multi-agent robotic systems are increasingly operating in real-world environments. The actions of these systems have a tangible and significant impact, particularly so when operating in close proximity to humans. While we expect such systems to exhibit safe and accurate behavior, errors are inevitable, and in such circumstances it is vitally important that the agents are able to explain their behavior to human operators. Operators can then ascertain whether the agent is operating erroneously – thus requiring intervention – or correctly but in a non-obvious manner.

However, state-of-the-art multi-agent systems are often controlled by deep neural network models trained with reinforcement learning techniques [1]. While these methods have shown great ability to generate effective and generalizable models, they do so at the expense of interpretability, and the models often remain inscrutable to human operators [2]. This poses a significant risk, especially in end-to-end models, where a rationale cannot be readily determined for *why* a model produced a certain decision, let alone provide a mechanism for intervening and correcting the rationale should it be incorrect. Such a mechanism is particularly important for robotic systems where we often encounter shifts in data distributions, such as when transferring policies from simulated environments to the real world, leading to model errors. These errors are exacerbated in multi-agent systems, where errors in each individual agent are compounded and produce large errors in environment dynamics.

In this paper, we propose a general method for learning interpretable policies – concept policy models – for multi-agent reinforcement learning (MARL). Our approach is predicated on the insight that we can leverage domain knowledge from an expert in order to regularize the model and influence what information is encoded from observations. We organize this domain knowledge into a set of interpretable *concepts* and enforce the constraint that the model is able to predict these concepts from observations, after which the concepts are used to predict policy actions. Concepts are semantically meaningful labels that can be extracted from observations, such as the presence of a concrete or abstract feature in an observation, e.g., the existence of a tree or the intention of a human. Crucially,

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

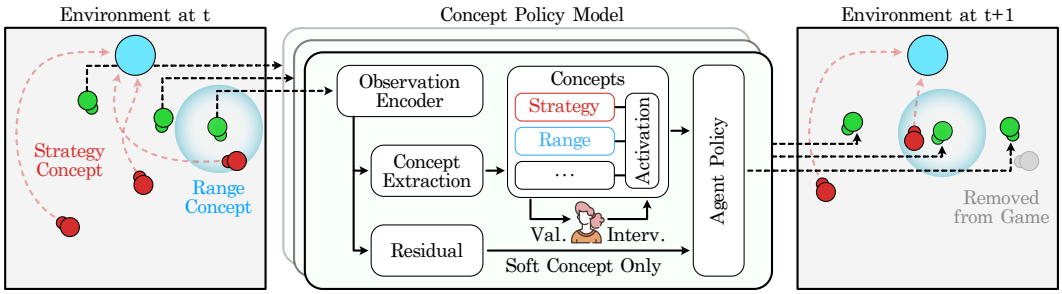

Figure 1: Concept policy models predict a set of interpretable concepts from observations, which are then used along with an (optional) residual to predict a policy action. A domain expert may intervene and provide corrective concept values to the policy if mis-predicted.

we find that the regularization imposed by the concept information helps stabilize the training process, and as a result leads to improved performance and sample efficiency.

A typical end-to-end neural network policy model maps observations to actions. Our approach inserts an intermediate *concept* layer, as shown in Fig. 1 which is required to predict concepts from observations. While this yields an interpretable model [3], it also imposes the assumption that the set of concepts are sufficient for policy inference. To ease this constraint, we introduce a scalable *residual* layer which passes additional information to the subsequent policy layers while ensuring it remains decorrelated with the concepts. We posit that the interpretability of the model is proportional to the capacity of the residual layer; intuitively, the more residual information available, the less the model may rely on the concepts. We show that this can result in a trade-off between interpretability and accuracy, given the expressivity of the concepts. Our contributions are as follows, we

- Introduce a general method for learning concept policy models in MARL utilizing expert domain knowledge, enabling a human operator to understand a policy's decision rationale and improving accuracy, sample efficiency, and training stability.
- Develop two specific formulations based on this method: soft-concept models and hard-concept models, and empirically show the trade-off between accuracy and interpretability.
- Formulate an intervention methodology and show how this can be used to offset model errors in general and in transfer learning (sim-to-real) scenarios.
- Empirically show that our proposed approach produces interpretable, intervenable MARL policy models which exceed the accuracy of baseline MARL policies in a simulated and real-world game of "tag", between two teams of 2, 3, and 5 robots each.

2 Related Work

Interpretability in supervised learning: Intepretability has been extensively studied within the field of supervised learning [4], which can be largely grouped into two categories: the explicit creation of an intrinsically interpretable model, or the post-hoc transformation of an uninterpretable model to an interpretable one. The former case typically revolves around considering interpretable classes of models – decision trees [5, 6], linear models [7, 8], or rule-based methods [9, 10] for example – and developing algorithms for these models. In the latter case, uninterpretable models are either transformed into interpretable ones [11, 12, 13], or interpretable models are extracted from an uninterpretable model for the purpose of explaining a model's decision rationale [14, 15].

Concept models often fall into the transformation case, and have been studied within the context of transforming a set of uninterpretable feature embeddings into a set of interpretable concepts [3, 16, 17]. A recent approach [18] similarly uses concepts, but rather than directly predicting such concepts it attempts to align the internal model representation to coincide with them.

Interpretability in reinforcement learning: As in supervised learning, interpretability for reinforcement learning largely falls into the two categories of intrinsically interpretable models and post-hoc

transformations. However, there is an additional line of work in which methods are devised to explain aspects particular to the Markov decision process model employed by RL methods. Some approaches have focused on interpretable representation learning [19, 20] and hierarchical decompositions [21], while others have opted to tackle MDP-specific explanations such as an interpretable reward signal [22] or action explanation [23, 24]. However, to the best of our knowledge, concept-related methods have not yet been explored in an RL setting, let alone MARL.

3 Preliminaries

Multi-Agent Reinforcement Learning: We model the MARL problem as a decentralized partially observable Markov decision process (Dec-POMDP) [25]. A Dec-POMDP is defined as a tuple $\langle \mathcal{S}, \mathcal{U}, P, R, \mathcal{Z}, O, n, \gamma \rangle$ in which $\mathcal{S}$ is the state space, $\mathcal{U}$ shared action space, P the state transition function, R the shared reward function, $\mathcal{Z}$ the observation space, O the observation function, n the number of agents, and γ the discount factor. For a given time step, the environment has a state $\mathbf{s} \in \mathcal{S}$ and each agent $a \in \{a_0, \ldots, a_n\}$ samples a partial observation $\mathbf{z}_a \in \mathcal{Z}$ according to the observation function $O(s, a) \in \mathcal{Z}$. The agents simultaneously sample an action $\mathbf{u}_a \in \mathcal{U}$ inducing a state transition according to $P(\mathbf{s}'|\mathbf{s}, \mathbf{u}) \in [0, 1]$. Each agent receives a reward r_a according to the shared reward function $R(\mathbf{s}_a, \mathbf{u}_a) \in \mathbb{R}$ with a discount factor $\gamma \in [0, 1]$. We follow a centralized training and decentralized execution approach (CTDE), thus learning a central policy $\pi_\theta(\mathbf{u}_a|\mathbf{z}_a) \in [0, 1]$ parameterized by θ by maximizing the discounted expected cumulative reward: $\mathbb{E}_t[\sum_t \gamma^t R(\mathbf{s}_a, \mathbf{u}_a)]$.

Multi-Agent Proximal Policy Optimization: Multi-Agent Proximal Policy Optimization (MAPPO) [26] is a straightforward extension to standard PPO [27] under the CTDE assumption in which we learn a single actor, π_θ, and a single critic, V_ϕ, parameterized by θ and ϕ respectively. When sampling from the environment, each agent executes the same learned policy with their individual observations and actions. As with all policy gradient methods, PPO seeks to compute the policy gradient by differentiating the following objective function:

$$L(\theta) = \hat{\mathbb{E}}_t[\log \pi_\theta(\mathbf{u}_a|\mathbf{z}_a)\hat{A}_t], \tag{1}$$

where $\hat{A}$ is the estimated advantage function. PPO extends this objective function by adaptively clipping the update gradient and applying an entropy bonus to the policy to encourage exploration. If the value function and policy function share parameters, i.e., $\theta = \phi$, then the objective function must also include the value function loss.

4 Concept Policy Models

We propose a method for learning concept policy models, which integrates domain knowledge from an expert in the form of concepts into a neural network policy model. These concepts are intended to serve two purposes: they are useful predictors for the desired policy behavior and as such can serve as the basis of a decision rationale, and they can be corrected in real-time by an expert if mis-predicted by the policy in order to induce correct behavior. The expert provides an oracle function $V(\cdot)$ which can be used to predict a ground truth concept vector of size j given an observation from an agent a, $\mathbf{v} = V(\mathbf{z}_a)$ where $\mathbf{v} \in \mathbb{R}^j$. Concepts may be either continuous or discrete, and represent interpretable features which are assumed to be relevant to the task at hand. As an example, let us consider a cooperative-competitive multi-agent game in which two teams of agents play a game of "tag" during which one side must prevent the other from reaching a specific location. In this game, an expert might identify a specific feature such as the relative location of the nearest opposing team member, or a more abstract feature such as the opposing team's strategy as a concept.

4.1 Policy Concept and Residual Layers

We integrate this concept information into an end-to-end neural network policy model $\pi_\theta(\mathbf{u}_a|\mathbf{z}_a)$ which predicts the probability for agent a taking action $\mathbf{u}_a$ given the observation $\mathbf{z}_a$ and parameters θ. This is accomplished by inserting an intermediate layer $c_{\theta c}(\cdot)$ of size j into the network to estimate the concept vector, dividing the network into two parts: $\pi^1_{\theta 1}(\cdot)$ representing the portion of the network before the new layer, and $\pi^2_{\theta 2}(\cdot)$ representing the portion of the network after the new layer, such that

$$\pi(\mathbf{u}_a|\mathbf{z}_a) = \pi^2_{\theta 2}(c(\mathbf{x}) + r(\mathbf{x})) \quad \text{where} \quad \mathbf{x} = \pi^1_{\theta 1}(\mathbf{u}_a|\mathbf{z}_a) \tag{2}$$

and $r(\cdot)$ is a residual layer of size k designed to pass through non-concept information and the concept layer acts as a concept predictor, such that $\hat{v} = c(\mathbf{x})$. In our proposed concept policy model, $\pi^1_{\theta 1} : \mathbb{R}^{|\mathbf{z}|} \rightarrow \mathbb{R}^h$ acts as a feature encoder mapping an observation to a feature embedding. The newly inserted concept layer serves as a bottleneck such that $c(\cdot) : \mathbb{R}^h \rightarrow \mathbb{R}^j$ maps the feature embedding to a concept vector, while the residual layer $r(\cdot) : \mathbb{R}^h \rightarrow \mathbb{R}^k$ maps the feature embedding to a residual vector. The final policy layer $\pi^2_{\theta 2}(\cdot) : \mathbb{R}^{j+k} \rightarrow \mathbb{R}^{|\mathbf{u}|}$ maps the aggregated concept and residual vectors to a policy action. We train the concept layer $c(\cdot)$ by imposing an additional auxiliary loss $L^c(\theta)$ in the objective function optimized by MAPPO:

$$
\begin{aligned}
L(\theta) &= \hat{\mathbb{E}}_t[\log \pi_\theta(\mathbf{u}_a|\mathbf{z}_a)\hat{A}_t] - L^c(\theta) \quad \text{where,} \\
L^c(\theta) &= \sum_{i=0}^{j} L^c_i(\theta) \quad \text{and} \quad L^c_i(\theta) = \begin{cases} \text{FL}(v_i, \hat{v}_i) & \text{if discrete} \\ \text{MSE}(v_i, \hat{v}_i) & \text{if continuous.} \end{cases}
\end{aligned} \tag{3}
$$

This loss is summed over each concept: mean squared error (MSE) is used for continuous concepts, and focal loss (FL) [28] for discrete concepts. The focal loss is a cross-entropy variant designed for class imbalanced situations which are likely to occur in our concept setting, as some concepts may be significantly rarer than others. In our above example with the strategy concepts, some strategies may be much less likely to occur than others, for instance. Note that for multi-class discrete concepts, a single abstract concept may consist of multiple nodes, and we refer to this as a *concept group*. In the strategy case, suppose an agent team may only execute one of strategy A, B, or C at a time, thus these three concepts represent a single concept group and so when we pass the discrete concepts through a softmax activation in order to calculate the focal loss we do so in a group-wise manner.

The goal of the residual layer is to pass through information from $\pi^1_{\theta 1}(\cdot)$ that is not captured by the concept vector. Without the residual, the concept vector must sufficiently represent the observation so that $\pi^2_{\theta 2}$ accurately infer the agent's action from concepts alone (a strict assumption in practice). We define two concept policy model variants: **hard concept policy models** which contain no residual ($k = 0$), and **soft concept policy models** which do ($k > 0$).

By examining the concept layer activations, a human operator can query the predicted concepts $\hat{v}$ and understand what concepts the policy model used for prediction. However, we conjecture that there is an inherent trade-off between the size of the residual layer k and the interpretability of these activations. While a full interpretability analysis is outside the cope of this work, we posit that the greater the residual dimension, the less that $\pi^2_{\theta 2}(\cdot)$ must rely on the concept vector, i.e., there is a larger amount of non-concept information on which to base its prediction – which follows that k is inversely proportional to interpretability.

4.2 Concept and Residual Whitening

In order to constrain the residual $r(\cdot)$ such that it does not encode information related to the concepts, we decorrelate the neuron activation vectors via whitening. Given a matrix $\mathbf{X} \in \mathbb{R}^{b \times j+k}$ consisting of the activations from the concatenated concept and residual vectors over a mini-batch of b samples, we aim to produce a whitened matrix $\mathbf{X}'$ with ZCA whitening via iterative normalization [29]

$$
\mathbf{X}' = \mathbf{D}\Lambda^{-\frac{1}{2}}\mathbf{D}^T(\mathbf{X} - \mu_x) \tag{4}
$$

where $\mathbf{D}$ and Λ are the eigenvectors and eigenvalues of $\mathbf{X}$ respectively. Iterative normalization uses an iterative optimization technique to incrementally whiten the matrix $\mathbf{X}$, where the hyperparameter T dictates the number of optimization iterations. This gives us the flexibility of only partially decorrelating the residual and whitening layers, if desired, by setting T to a smaller value, e.g., $T = 2$. In practice, we find that performing fewer iterations is often necessary to stabilize the training process, as a higher T tends to increase the stochastic normalization disturbance and leads to reduced training efficiency [29], which is particularly noticeable in a MARL setting. At each training iteration, we first perform whitening then backpropagate our computed gradients, thus allowing us to decorrelate the concept and residual layers without requiring an additional optimization step as in prior work [18].

	Setup	Model	WR	Intervened WR	*Range*	*Strategy*	*Target*	*Orientation*	*Position*
Simulation	2v2	Soft	51%	55%	0.03	0.04	0.24		
		Hard	83%	84%	0.04	0.07	0.2	0.10	0.11
		Base	34%						
	3v3	Soft	55%	57%	0.03	0.10	0.17		
		Hard	74%	80%	0.03	0.13	0.23	0.11	0.14
		Base	16%						
	5v5	Soft	32%	40%	0.02	0.25	0.48		
		Hard	78%	86%	0.03	0.14	0.13	0.11	0.21
		Base	31%						
Real	2v2	Soft	10%	0%	0.02	0.88	0.02		
		Hard	25%	95%	0.04	0.81	0.01	3.33	0.08
		Base	35%						

Table 1: The win rate (WR) and concept errors for our proposed models (Soft and Hard) and a baseline without concepts (Base). The Hard model is trained over all concepts, the Soft model over a subset, and the Base model with none. The Win Rate is the standard win rate of the policy when the policy is executed, while the Intervened Win Rate (Intervened WR) is the win rate when an expert intervenes over all concepts. *Range*, *Strategy*, and *Target* are discrete concepts and as such the error shown is the error in accuracy score, while *Orientation* and *Position* are continuous and indicate mean squared error. *Orientation* is in radians and *Position* is a unit-less value in $[-1, 1]$.

4.3 Policy Intervention

In addition to querying the predicted concepts $\hat{v}$, a human operator may also decide to intervene when these predictions are incorrect. This can be achieved by explicitly overwriting the concept layer node activations (or softmax activations for discrete concepts) with the appropriate values. We denote the modified concepts as $\bar{v}$ which leads to the following intervened concept policy model: $\pi_\theta(\mathbf{u}_a|\mathbf{z}_a, \bar{v}) = \pi^2_{\theta 2}(\bar{v} + r(\mathbf{x}))$. This policy intervention corrects prediction errors in the feature encoder $\pi^1_{\theta 1}(\cdot)$. We find that these sorts of errors are particularly prevalent when transferring robot policies from simulation to the real world due to different observation distributions, and show that policy interventions are effective at reducing such errors in Sec. 6. Similar to interpretability, we hypothesize that intervention effectiveness is inversely proportional to the size of the residual k; we cannot intervene on the residual layer activations so any resulting errors will persist. While our empirical results hint in this direction, we save a full exploration for future work.

5 Experimental Setup

We show that our proposed concept policy models achieve high policy success rates and concept accuracy, in addition to improved training stability and sample efficiency, over standard MARL models in a cooperative-competitive multi-agent game of "tag" previously described in Sec. 4. We empirically analyze our approach in both simulated and real-world versions of this game, and explore its strengths and weaknesses especially with respect to interventions and sim-to-real transfer in Sec. 6.

5.1 Tag Game

In our game, two teams of agents compete with each other in which one team attempts to reach a specified goal location while the other team defends it and attempts to keep them away, which we refer to as the attacking and defending team respectively. An agent from each team may "tag" an agent from the opposing team as long as it lies within a given proximity and is facing the opposing agent, removing the tagged agent from play. The attacking team wins if any agent is able to reach the goal location, while the defending team wins if the attacking agents are all tagged or the maximum number of time steps elapses. Each team consists of the same number of agents, and we vary this number between 2, 3, and 5. This environment allows for complex behaviors in which attackers and defenders may employ coordinated strategies in order to out-maneuver the other team to win, and provides a suitable scenario for testing both specific and abstract concepts.

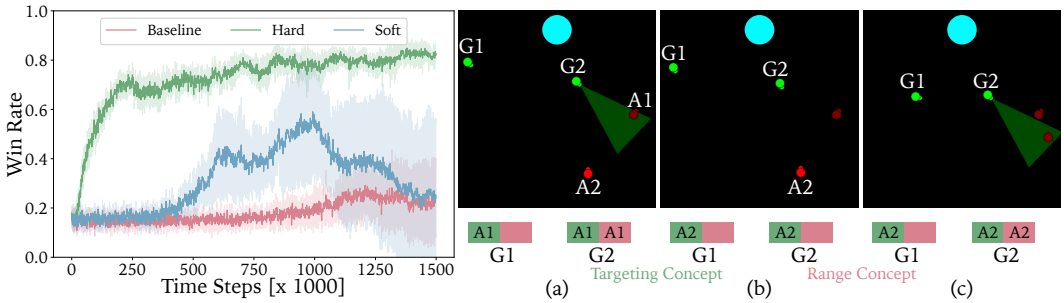

Figure 2: Left: training curves showing win rate vs iterations over 5 random training seeds for each tested model type in a 2v2 scenario. Right: a sequence of episode steps showing the concept activations for agents on the defending team (green).

Observation and Action Space: The observations are a set of extracted features consisting of the positions, velocities, orientations and tagged status of all agents. Actions consist of accelerating forward or backward by a fixed amount, rotating left or right by a fixed offset, and tagging.

Reward: Our environment is a modified version of FortAttack [30]. We have simplified the reward function by removing penalties to encourage policy exploration, resulting in a reward of the form

$$R(s_a, u_a) = -R_{\text{Ori.}} - R_{\text{Miss}} - R_{\text{Tagged}} - R_{\text{Lose}} + R_{\text{Tag}} + R_{\text{Win}}, \tag{5}$$

where $R_{\text{Ori.}}$ is a penalty for not facing an opponent, R_{Miss} is a penalty for missing a tag, R_{Tagged} is a penalty for being tagged, R_{Lose} is a penalty for losing, R_{Tag} is a reward for tagging, and R_{Win} is a reward for winning. Several of these are shaping terms in order to improve sample efficiency, which we found to be necessary for efficient convergence rates in the absence of expert demonstrations.

Strategy: Furthermore, we restricted our game such that only the defending team's policy is trained via MARL. While it is a straightforward extension to train both an attacker and defender policy iteratively, we opted to restrict the attacking team to sampling strategies from a fixed policy distribution to better investigate the effects of our concept policy model on performance. We sample attacker strategies from a distribution consisting of three "types" with equal probabilities, $\{$*random*, *left*, *right*$\}$, where the attackers execute random actions, move towards the goal by sweeping along the left side of the environment, and move towards the goal by sweeping along the right, respectively. Given a sampled team level strategy, each agent then sampled an individual policy with noise from a further distribution, so as to generate stochastic policies.

Concepts: We utilized the following concepts: $\{$*Range*, *Strategy*, *Target*, *Orientation*, *Position*$\}$, in which *Range* is a boolean concept indicating whether the opposing agent specified by *Target* is within tagging range, *Strategy* is a categorical concept mapping to the above team-level strategies, *Target* is a categorical concept indicating an opposing agent that should be pursued, and *Orientation* and *Position* are continuous concepts encoding the relative orientation and position of each opposing agent, respectively. The hard concept policy models are trained with the full set of concepts, while the soft models only employ a subset consisting of $\{$*Range*, *Strategy*, *Target*$\}$.

Real-world Equivalent: The real-world version of our tag game is played in a 2v2 scenario on a $6' \times 6'$ play area, with four Khepera IV [31] robots. Policies are trained in the simulation environment, then executed in the real-world environment; no additional training and no few-shot conditioning is employed. The robot positions and orientations are extracted from a Vicon [32] motion capture system and converted into the model's expected observation format. The real-world version of the game exhibits significant differences in the dynamics between the simulated robots and the real-world robots – particularly in velocities, accelerations, and even control – presenting a challenging environment for sim-to-real. Further, tagged agents disappear in simulation while the real-world agents are driven out of the play area, providing a temporary obstacle.

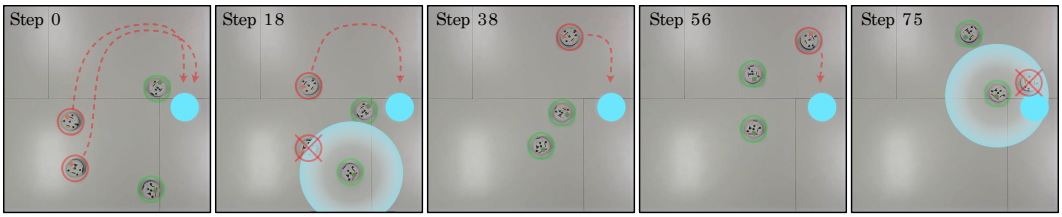

Figure 3: A sequence of steps from a single episode during a policy execution in the real-world. The blue circle is the attacking team's (red) goal while the defending team (green) attempts to stop them.

5.2 Concept Policy Models and Baselines

We trained a hard and soft concept model for 10M time steps for each scenario – 2v2, 3v3, and 5v5 – along with a standard policy model without concepts. Each model consists of a series of fully connected layers, recurrent layers, and the iterative normalization layer applied over the concatenated concept and residual layers, with full details given in the supplementary material. The concept dimension j for each hard model differ for each scenario due to the number of agents: $j = 13$, $j = 18$, and $j = 28$ for 2v2, 3v3, and 5v5 respectively. The concept dimensions for the soft models are $j = 9$, $j = 12$, and $j = 18$ for 2v2, 3v3, and 5v5. For the soft models, we additionally provide a residual layer with dimension $k = 23$, $k = 52$, $k = 78$ for 2v2, 3v3, and 5v5, respectively, leading to a combined bottleneck size of 32, 64, and 96. The baseline model lacks a concept layer ($j = 0$) and has a full-width residual $k = 128$. Residual layers sizes and other hyperparameters are given in the supplementary material and were chosen through extensive hyperparameter optimization.

6 Results

The win rates and concept accuracy errors for the defending team in both simulation and real-world are shown in Table 1. These values are computed by training two seeds with the best set of hyperparameters found during optimization, then rolling out each policy for 100 evaluation episodes in simulation, and 20 in real-world.

Simulation: We first observe that both concept policy model variants out-perform the baseline model in each scenario, with the hard concept model outperforming the others by a large margin. This in itself is unsurprising, given that the concepts were hand-designed so as to provide a sufficient amount of information for the policy, and the hard concept policy model heavily regularizes the learned model such that it learns this information. The decreased performance in the soft model is due to the fact that it is only trained with a subset of concepts, notably lacking both the *Orientation* and *Position*, and consequently the residual struggles to fully encode this information on its own. Due to the importance of these concepts, the performance suffers. The baseline model also significantly under-performs both of our proposed models, indicating that it struggles to learn appropriate features in the absence of regularization. The intervened win rate follows when an expert intervenes and sets the correct concept value when a concept is incorrectly predicted by the model. This results in an improved win rate, particularly in the 3v3 and 5v5 scenarios, which is reasonable given that these are more complex environments, and as can be seen in Table 1, yield increased concept errors. Figure 2 shows the training curves for the three model types in a 2v2 scenario and clearly demonstrate the improved training stability offered by the hard concept model. Conversely, the soft concept models exhibit increased performance over the baseline models at the cost of decreased stability.

Real-world: In the real-world environment shown in Fig. 3, we can immediately see that the win rates are drastically reduced for the concept policy models, but surprisingly not for the baseline model. Qualitatively, we have observed that this is because the baseline model became trapped in a local minima and learned a policy which was semi-performant and independent of the actions of the opposing team. The baseline policy resulted in the defenders driving in circles around the goal while continuously tagging. This is clearly a sub-optimal behavior as it only reaches a 40% win rate in both simulation and the real-world, but it indicates the difficulty which standard MARL policies face when attempting to learn meaningful feature embeddings. The other interesting result from this experiment

is the massive gain in performance by the hard model when interventions occur, and similarly the complete lack of improvement when the soft model is intervened. We can draw two insights from this: the distribution shift from the simulated to the real world environment is largely contained within the feature extractor, which is compensated for by the interventions in the hard model; and that the *Orientation* and *Position* concepts are by far the most important as when we are unable to intervene on them and correct for dynamics errors as in the soft model, performance fails to improve.

Concept Ablations: Next, we examine ablated hard concept policy models which are only trained over a subset of concepts in the 2v2 simulated scenario. The results are shown in Table 2 and further support the evidence that the *Orientation* and *Position* concepts are by far the most important with respect to the win rate. In the simulation environment, the win rate drops to 23% in the absence of those concepts, while in the real-world environment it drops to 27%. Note that the only difference between the hard model and the soft model with the RST concept set is the presence of a residual layer; when this residual is present and allowed to encode additional information the win rate is nearly doubled to 51% as in the soft model in Table 1.

	2v2 Win Rate	3v3 Win Rate
All	83%	74%
RST	23%	27%
RTOP	80%	69%
OP	80%	72%
	Simulation Win Rate	**Real Win Rate**
All	84%	95%
RST	84%	20%
RTOP	81%	50%
OP	78%	80%

Table 2: Top: Concept ablations in the 2v2 and 3v3 simulated scenario when only a subset of concepts are trained. Bottom: Intervention ablations in the 2v2 scenario when only a subset of concepts are intervened over. All consists of all concepts, RST = {*Range*, *Strategy*, *Target*}, RTOP = {*Range*, *Target*, *Ori.*, *Pos.*}, and OP = {*Ori.*, *Pos.*}.

Intervention Ablations: We performed an ablation over the set of intervened concepts, with the results shown in Table 2. We can first observe that ignoring interventions over the Orientation and Position concepts does not affect the win rate, likely because the associated errors for those concepts is already low as shown in Table 1. As we intervene over fewer and fewer concepts, the win rate further drops; however, paradoxically the win rate drops to below the base win rate without any interventions at all. In the real-world we observe the opposite effect where intervening over a larger set of concepts always improves the win rate. This is especially so for the *Orientation* and *Position* concepts which account for a nearly 45% increase in the win rate alone, allowing us to conclude that not only are these concepts important, but that the distribution shift in the observation data from simulation to real particularly affects this concept.

Limitations: In order to analyze the performance of our model for a single team of agents, we have restricted the variability in our environment and reduced the complexity of possible behaviors. In the future, we would like to evaluate asymmetric team compositions and learn a policy for the attackers. We have also only considered low-dimensional inputs in our experiments, and although we expect our approach to scale well to rich input representations such as images, since concept models have been traditionally applied in vision domains, this remains an open question. Additionally, we would like to expand the complexity of our real-world environments by incorporating additional robots.

7 Conclusion

In this work we have introduced *concept policy models* for Multi-Agent Reinforcement Learning which incorporate domain knowledge from an expert in the form of concepts. We have developed a general framework in which concepts may be optionally augmented with residual information in order to ease the restriction that they fully express the information necessary for policy prediction, and show that this results in concept policy models which fall along a spectrum of regularization: hard concept models in which no residual is allowed, and soft concept models in which it is. We empirically show that this regularization greatly stabilizes training and results in improved accuracy and sample efficiency, and crucially, allows a human operator to query the model for its concept activations which provide an interpretable *rationale* for the policy's decisions. We further show that when the operator intervenes and corrects incorrect concept predictions, we can improve policy accuracy and partly compensate for distribution shifts, particularly in sim-to-real transfer scenarios.

References

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

[2] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

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

[4] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[5] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[7] S. Haufe, F. Meinecke, K. Görgen, S. Dähne, J.-D. Haynes, B. Blankertz, and F. Bießmann. On the interpretation of weight vectors of linear models in multivariate neuroimaging. *Neuroimage*, 87:96–110, 2014.

[8] T. Le Nguyen, S. Gsponer, I. Ilie, M. O'Reilly, and G. Ifrim. Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. *Data mining and knowledge discovery*, 33(4):1183–1222, 2019.

[9] X. Yin and J. Han. Cpar: Classification based on predictive association rules. In *Proceedings of the 2003 SIAM international conference on data mining*, pages 331–335. SIAM, 2003.

[10] F. Wang and C. Rudin. Falling rule lists. In *Artificial intelligence and statistics*, pages 1013–1022. PMLR, 2015.

[11] R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. E. Hinton. Neural additive models: Interpretable machine learning with neural nets. *Advances in Neural Information Processing Systems*, 34:4699–4711, 2021.

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

[13] G. Liu, O. Schulte, W. Zhu, and Q. Li. Toward interpretable deep reinforcement learning with linear model u-trees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 414–429. Springer, 2018.

[14] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*, 2018.

[15] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. 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.

[16] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31, 2018.

[17] M. Losch, M. Fritz, and B. Schiele. Interpretability beyond classification output: Semantic bottleneck networks. *arXiv preprint arXiv:1907.10882*, 2019.

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

[19] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

[20] Z. Lin, K.-H. Lam, and A. Fern. Contrastive explanations for reinforcement learning via embedded self predictions. *arXiv preprint arXiv:2010.05180*, 2020.

[21] B. Beyret, A. Shafti, and A. A. Faisal. Dot-to-dot: Explainable hierarchical reinforcement learning for robotic manipulation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5014–5019. IEEE, 2019.

[22] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI Workshop on explainable artificial intelligence*, 2019.

[23] F. Cruz, R. Dazeley, P. Vamplew, and I. Moreira. Explainable robotic systems: Understanding goal-driven actions in a reinforcement learning scenario. *Neural Computing and Applications*, pages 1–18, 2021.

[24] H. Yau, C. Russell, and S. Hadfield. What did you think would happen? explaining agent behaviour through intended outcomes. *Advances in Neural Information Processing Systems*, 33: 18375–18386, 2020.

[25] F. A. Oliehoek and C. Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

[26] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu. The surprising effectiveness of mappo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

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

[28] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[29] L. Huang, Y. Zhou, F. Zhu, L. Liu, and L. Shao. Iterative normalization: Beyond standardization towards efficient whitening. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4874–4883, 2019.

[30] A. Deka and K. Sycara. Natural emergence of heterogeneous strategies in artificially intelligent competitive teams. In *International Conference on Swarm Intelligence*, pages 13–25. Springer, 2021.

[31] J. M. Soares, I. Navarro, and A. Martinoli. The khepera iv mobile robot: performance evaluation, sensory data and software toolbox. In *Robot 2015: second Iberian robotics conference*, pages 767–781. Springer, 2016.

[32] Vicon Motion Systems. Vicon motion capture system. URL https://www.vicon.com/hardware/cameras/.