LICORICE: LABEL-EFFICIENT CONCEPT-BASED INTERPRETABLE REINFORCEMENT LEARNING

Zhuorui Ye[*][†]
Institute for Interdisciplinary Information Sciences
Tsinghua University
Beijing, China
cuizhuyefei@gmail.com

Stephanie Milani[*]
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
smilani@cs.cmu.edu

Geoffrey J. Gordon
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
ggordon@cs.cmu.edu

Fei Fang
Software and Societal Systems Department
Carnegie Mellon University
Pittsburgh, PA 15213
feifang@cmu.edu

ABSTRACT

Recent advances in reinforcement learning (RL) have predominantly leveraged neural network policies for decision-making, yet these models often lack interpretability, posing challenges for stakeholder comprehension and trust. Concept bottleneck models offer an interpretable alternative by integrating human-understandable concepts into policies. However, prior work assumes that concept annotations are readily available during training. For RL, this requirement poses a significant limitation: it necessitates continuous real-time concept annotation, which either places an impractical burden on human annotators or incurs substantial costs in API queries and inference time when employing automated labeling methods. To overcome this limitation, we introduce a novel training scheme that enables RL agents to efficiently learn a concept-based policy by only querying annotators to label a small set of data. Our algorithm, LICORICE, involves three main contributions: interleaving concept learning and RL training, using an ensemble to actively select informative data points for labeling, and decorrelating the concept data with a simple strategy. We show how LICORICE reduces human labeling efforts to 500 or fewer concept labels in three environments, and 5000 or fewer in two complex environments, all at no cost to performance. We also explore the use of VLMs as automated concept annotators, finding them effective in some cases but challenging in others. This work significantly reduces the annotation burden for interpretable RL, making it more practical for real-world applications that necessitate transparency. Our code is released.[1]

1 INTRODUCTION

In reinforcement learning (RL), agents learn a *policy*, which is a strategy for making sequential decisions in complex environments. Agents typically represent the policy as a neural network, as this representation tends to yield high performance (Mirhoseini et al., 2021). However, this choice can come at a cost: such policies are challenging for stakeholders to interpret — particularly when the network inputs are also complex, such as high-dimensional sensor data. This opacity can pose a significant hurdle, especially in applications where understanding the rationale behind decisions is critical, such as healthcare (Yu et al., 2021) or finance (Liu et al., 2022). In such applications, decisions can have significant consequences, so it is essential for stakeholders to fully grasp the reasoning behind actions to confidently adopt or collaborate on a policy.

---

[*] Equal Contribution.

[†] This work was done when Ye was a visiting intern at CMU.

[1] https://github.com/cuizhuyefei/LICORICE

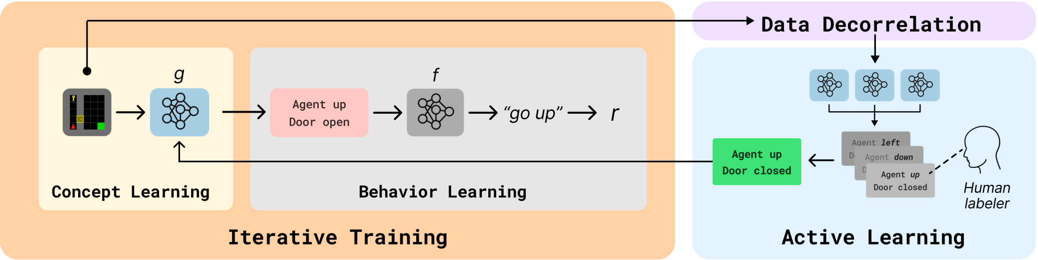

Figure 1: **LICORICE overview.** In concept-based RL, the policy includes a bottleneck to map from states to concepts with g, and then from concepts to (distributions over) actions with f. During training, LICORICE addresses concept label efficiency concerns with three key components: i) iterative training, ii) data decorrelation, and iii) active learning.

To address interpretability concerns in supervised learning, recent works have integrated human-understandable concepts into neural networks through concept bottleneck models (Koh et al., 2020; Espinosa Zarlenga et al., 2022). These models insert a bottleneck layer whose units correspond to interpretable concepts, ensuring that the final decisions depend on these concepts instead of on opaque raw inputs. By training the model both to have high task accuracy and to accurately match experts' concept labels, these models learn a high-level concept-based representation that is simultaneously meaningful to humans and useful for machine learning tasks. As an example, a concept-based explanation for a bird classification task might include a unit that encodes the bird's wing color. Concepts also offer a path to compositional generalization by enabling subtask decomposition into meaningful, reusable components (Mao et al., 2022; Wang et al., 2023).

More recently, these techniques have been applied to RL by incorporating a concept bottleneck in the policy (Grupen et al., 2022; Zabounidis et al., 2023), so that the chosen actions are a function of the human-understandable concepts. However, a significant challenge emerges in this method's practical application: past work assumes that concept annotations are readily available during RL training. To learn a mapping from states to concepts, an RL agent requires concept information for every state encountered during training, which is often millions or billions of state-action pairs. Many real-world domains—such as autonomous driving—are not accompanied by high-level concepts to support human-understandable decision-making. On one hand, relying on human labelers is impractical, risking errors due to fatigue (Marshall & Shipman, 2013). On the other hand, using vision language models (VLMs) for automated concept extraction (Oikarinen et al., 2023), while alleviating the human bottleneck, introduces significant API or inference costs that can be prohibitive.

In this work, we address this challenge with LICORICE (**L**abel-efficient **I**nterpretable **CO**ncept-based **ReI**nfor**CE**ment learning), a novel training paradigm designed to minimize the number of concept annotation queries while maintaining high task performance. Figure 1 illustrates our algorithm. LICORICE tackles three key challenges. First, it addresses the problem of concept learning on off-policy or outdated data. If concepts are learned from data collected by a random policy, the concept distribution may not reflect the distribution under an optimal policy. To ensure that concept learning occurs on more recent and on-policy data, LICORICE interleaves concept learning and RL training through *iterative training*: it alternately freezes the network layers corresponding to either the concept learning part or the decision-making part. Second, LICORICE addresses the problem of limited training data diversity that occurs when an agent interacts with the environment, thereby generating sequences of highly similar, temporally correlated data points. To tackle this issue, LICORICE implements a *data decorrelation* strategy to produce a more diverse set of training samples. Third, LICORICE addresses the inefficient use of annotation effort, where labeled samples may provide redundant information. To resolve this problem, we employ *disagreement-based active learning* using a concept ensemble to select the most informative data points for labeling.

To evaluate the effectiveness of LICORICE, we conduct experiments in two scenarios—perfect human annotation and VLM annotation—on five environments with image input: an image-based version of CartPole, two Minigrid environments, and two Atari environments. First, under the assumption of perfect human annotation, we show that LICORICE yields both higher concept accuracy and higher reward while requiring fewer annotation queries compared to baseline methods. Second, we find that VLMs can indeed serve as concept annotators for some, but not all, of the environments.

Our contributions are summarized as follows:

- To the best of our knowledge, we are the first to investigate the problem of a limited concept annotation budget for interpretable RL. We introduce LICORICE, a novel training scheme that enables label efficient learning of concept-based RL policies.
- We conduct extensive experiments to show the effectiveness of LICORICE across five environments with varying budget constraints.
- We study the use of VLMs as automated concept annotators for concept-based RL, demonstrating their effectiveness in certain environments while highlighting challenges in others.

2 PRELIMINARIES

Reinforcement Learning. In RL, an agent learns to make decisions by interacting with an environment (Sutton & Barto, 2018). The environment is commonly modeled as a Markov decision process (Puterman, 2014), consisting of the following components: a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a state transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ that indicates the probability of transitioning from one state to another given an action, a reward function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ that assigns a reward for each state-action-state transition, and a discount factor $\gamma \in [0, 1]$ that determines the present value of future rewards. The agent learns a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, which maps states to distributions over actions with the aim of maximizing the expected cumulative discounted reward. We evaluate a policy via its value function, which is defined as $V^\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s]$. The ultimate aim in RL is to determine the optimal policy, π^*, through iterative refinement based on environmental feedback.

Concept Policy Models. Concept-based explanations have emerged as a common paradigm in explainable AI (Poeta et al., 2023). They explain a model's decision-making process through human-understandable attributes and abstractions. In supervised learning, concept bottleneck models (Koh et al., 2020) implement this approach using two key functions: a concept predictor $g : X \to C$, mapping inputs to interpretable concepts, and a label predictor $f : C \to Y$, mapping the concept predictions to a downstream task space, such as labels for classification. As a result, the prediction takes the form $\hat{y} = f(g(x))$, where the input x influences the output solely through the bottleneck $\hat{c} = g(x)$. In RL, we insert a concept bottleneck layer into the policy network:

$$\pi(s) = f(g(s)),$$

such that π maps from states $s \in \mathcal{S}$ to concepts $c \in C$ to actions $a \in \mathcal{A}$ (Grupen et al., 2022; Zabounidis et al., 2023). This setup allows the policy to base its decisions on understandable and meaningful features. As a result, we can use any RL algorithm that can be modified to include an additional loss function for concept prediction.

Training Concept Policy Models. Training these models requires a dataset of state-concept-action triplets $(s, c, a) \in \mathcal{S} \times C \times \mathcal{A}$. The functions f and g are typically implemented as neural networks, with their parameters collectively defined as θ. Previous work (Zabounidis et al., 2023) simply combines the concept prediction loss $L^C(\theta)$ and RL loss $L^{\text{RL}}(\theta)$:

$$L(\theta) = L^{\text{RL}}(\theta) + L^C(\theta),$$

where the exact definitions of $L^{\text{RL}}(\theta)$ and $L^C(\theta)$ depend on the choice of RL algorithm and concept learning task. The objective is to find the optimal parameters θ^* that minimize this combined loss function. However, this approach requires continuous access to ground-truth concepts for training f, which may not always be feasible or desirable in practical RL scenarios.

3 LICORICE

As we have mentioned, the standard way of training concept-based RL assumes continuous access to an oracle to provide concept labels. However, this assumption is problematic due to the large annotation cost incurred by human or automated labeling efforts. To reduce the number of concept labels required for concept-based RL, we propose LICORICE, a novel algorithm for interpretable RL consisting of three main components: iterative training, data decorrelation, and disagreement-based active learning. The full pseudocode is in Algorithm 1.

Algorithm 1 LICORICE (**L**abel-efficient **I**nterpretable **CO**ncept-based **ReI**nfor**CE**ment learning)

```
1: Input: Total budget $B$, number of iterations $M$, sample acceptance threshold $p$, ratio $\tau$ for active learning, batch size for querying $b$, number of concept models $N$ to ensemble
2: Initialize: training set $\mathcal{D}_{\text{train}} \leftarrow \emptyset$, and validation set $\mathcal{D}_{\text{val}} \leftarrow \emptyset$
3: for $m = 1$ to $M$ do
4:     Allocate budget for iteration $m$: $B_m \leftarrow \frac{B}{M}$
5:     while $|\mathcal{U}_m| < \tau \cdot B_m$ do
6:         Execute policy $\pi_m$ to collect unlabeled data $\mathcal{U}_m$ using acceptance rate $p$
7:     end while
8:     Initialize iteration-specific datasets for collecting labeled data: $\mathcal{D}'_{\text{train}} \leftarrow \emptyset$, $\mathcal{D}'_{\text{val}} \leftarrow \emptyset$
9:     while $B_m > 0$ do
10:        Train $N$ concept models $\{\tilde{g}_i\}_{i=1}^{N}$ on $\mathcal{D}_{\text{train}} \cup \mathcal{D}'_{\text{train}}$, using $\mathcal{D}_{\text{val}} \cup \mathcal{D}'_{\text{val}}$ for early stopping
11:        Calculate acquisition function value $\alpha(s)$ for all $s \in \mathcal{U}_m \setminus (\mathcal{D}'_{\text{train}} \cup \mathcal{D}'_{\text{val}})$
12:        Choose $b_m = \min(b, B_m)$ unlabeled points from $\mathcal{U}_m$ according to $\arg\max_s \alpha(s)$
13:        Query for concept labels to obtain new dataset $\mathcal{D}_m \leftarrow \{(s, c)\}^{b_m}$
14:        Split $\mathcal{D}_m$ into train and validation splits and add to $\mathcal{D}'_{\text{train}}$, $\mathcal{D}'_{\text{val}}$
15:        Decrement budget: $B_m \leftarrow B_m - b_m$
16:    end while
17:    Aggregate datasets: $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{train}} \cup \mathcal{D}'_{\text{train}}$, $\mathcal{D}_{\text{val}} \leftarrow \mathcal{D}_{\text{val}} \cup \mathcal{D}'_{\text{val}}$
18:    Continue training the concept network $g$ on $\mathcal{D}_{\text{train}}$, using $\mathcal{D}_{\text{val}}$ for early stopping
19:    Freeze $g$ and continue training $f$ using standard RL training to obtain $\pi_{m+1}$
20: end for
```

Iterative Training. As the agent's policy improves, the distribution of the visited states and their associated concepts changes. Consider an MDP where states are indexed. With a random (initial) policy, the agent tends to visit small-index states near the initial state, encountering only the concepts relevant to those states. However, as the policy improves, the agent explores higher-index states, leading to a shift in both the state and concept visitation distribution. If we exhaust all queries at the beginning of training, we risk training the model only on the concepts associated with small-index states from the random policy, potentially missing important concepts that emerge as the agent explores. We therefore propose iterative training to enable LICORICE to progressively refine its understanding of concepts as the policy improves. Iterative training consists of two parts: concept learning and behavior learning. Assume that we have access to an annotated concept dataset $\mathcal{D}_{\text{train}}$ generated by our current policy π_m. Then, *concept learning* focuses on training the concept portion of the network g on this dataset $\mathcal{D}_{\text{train}}$ (lines 17 to 19). *Behavior learning* involves freezing g and training the behavior portion of the network f with any standard RL algorithm with its associated loss L^{RL} (line 19). Upon completion of behavior learning, we obtain an updated policy π_{m+1}, which serves to collect more unlabeled concept data to help train g in the next iteration.

Data Decorrelation. In the previous section, we omitted that we do not obtain ground-truth concept labels when rolling out the current policy π_m. Instead, rollouts produce a dataset $\mathcal{U}_m$ of unlabeled states as candidates for querying for ground-truth concepts from an oracle. Given that consecutive states from policy rollouts tend to be highly similar, leading to redundant and inefficient datasets, this setup raises the question of how to collect diverse and informative data. Simply collecting all encountered states would give us long chains of nearly-identical samples, undermining the diversity we need for effective learning. To resolve this challenge, we introduce data decorrelation with two key components: a budget multiplier τ that sets $|\mathcal{U}_m| = \tau \cdot B_m$, and a per-state acceptance probability p. This random acceptance/rejection mechanism leverages a fundamental property of Markov processes—states become increasingly independent as their temporal distance grows according to the mixing time—allowing us to build more diverse datasets from our policy rollouts.

Disagreement-Based Active Learning. Equipped with $\mathcal{U}_m$, we are now prepared to select data points for querying for concept labels. The purpose of this stage is to make use of our limited labeling budget B_m to collect a labeled dataset $\mathcal{D}_m$ for training g. We propose to train a concept ensemble—consisting of N independent concept models—from scratch on the dataset of training points $\mathcal{D}_{\text{train}}$ that has been aggregated over all iterations (line 10). We use this ensemble to calculate the disagreement-based acquisition function $\alpha(\cdot)$, which determines whether each candidate in $\mathcal{U}_m$ ought to receive a ground-truth concept label (line 11). This function targets samples where prediction disagreement is highest among ensemble members, as these points often represent areas of uncertainty or decision boundaries where additional labeled data would be most informative. For $\alpha(\cdot)$, we use two different formulations, depending on the concept learning task. For concept classification, we use a query-by-committee (Seung et al., 1992) approach, in which we prioritize points with a high proportion of predicted class labels that are not the modal class prediction (also called the variation ratio (Beluch et al., 2018)):

$$\alpha(s) = 1 - \max_{c \in C} \frac{1}{N} \sum_{i=1}^{N} [\tilde{g}_i(s) = c].$$

Here, $\tilde{g}_i(s)$ is the concept prediction of the i-th model on state s. For concept regression, we directly use the estimate of variance across the concept models as a measure of disagreement:

$$\alpha(s) = \sigma^2(s) = \frac{1}{N-1} \sum_{i=1}^{N} (\tilde{g}_i(s) - \mu(s))^2,$$

where $\mu(s) = \frac{1}{N} \sum_{i=1}^{N} \tilde{g}_i(s)$ is the mean prediction of the N concept models. After querying for B_m ground-truth concept labels (line 13) using $\alpha(\cdot)$, we are prepared to continue training g during the concept learning stage.

Implementation Details. For training g (line 18), we employ two types of loss functions depending on the nature of the concepts: MSE for continuous concepts and cross-entropy loss for categorical concepts. If the problem requires mixed-type concepts, we either discretize continuous attributes or simply use a mixed loss. To train f (line 19), we freeze g and use PPO to continue training f from the previous iterations, resulting in the final policy π_{M+1}. For complex environments that require many iterations, we observe that RL-related parameters may get stuck in a local region in early iterations, becoming hard to optimize later. To mitigate this issue, in the last iteration, we additionally train a new RL network f' from scratch when fixing g with π_{M+1} as the anchoring policy to get the final π'_{M+1}. The anchoring policy ensures π'_{M+1} has a similar distribution to the previous one and the annotated observations are still highly relevant. Specifically, we initialize another policy π_θ with the same frozen g parameters and randomly initialized f, then train it with the standard PPO loss function and an additional KL-divergence penalty between the current policy and π_{M+1}:

$$L(\theta) = L^{\text{PPO}}(\theta) + \beta \cdot \text{KL}[\pi_{M+1}(\cdot \mid s), \pi_\theta(\cdot \mid s)].$$

4 EXPERIMENTS

In our experiments, we investigate the following questions:

RQ 1 Under a limited concept annotation budget, does LICORICE enable both high concept accuracy and high environment reward?

RQ 2 How effective are VLMs as automated concept annotators when used with LICORICE?

RQ 3 How label efficient is LICORICE compared with other methods?

RQ 4 Does LICORICE support test-time concept interventions?

4.1 EXPERIMENT SETUP

In each experiment, we run each algorithm 5 times, each with a random seed. All algorithms use PPO (Schulman et al., 2017; Raffin et al., 2021) with a concept bottleneck. More implementation details and hyperparameters are in Appendix A.2.

Environments. We investigate these questions across five environments: PixelCartPole (Yang et al., 2021), DoorKey (Chevalier-Boisvert et al., 2023), DynamicObstacles (Chevalier-Boisvert et al., 2023), Boxing (Bellemare et al., 2013), and Pong (Bellemare et al., 2013). Each environment includes a distinct challenge and features a set of interpretable concepts describing key objects and properties. We summarize the concepts in Table 1, with more details in Appendix A.1. These environments are characterized by their dual representation: a complex image-based input and a symbolic concept-based representation. PixelCartPole, DoorKey, and DynamicObstacles are simpler because we can extract noiseless ground-truth concept labels from their source code. In contrast,

Environment	# Concepts	Concept Type	Binarized # Concepts
PixelCartPole	4	Continuous	N/A
DoorKey	12	Discrete	46
DynamicObstacles	11	Discrete	30
Boxing	8	Discrete	1480
Pong	7	Discrete	1848

Table 1: Summary of environments, including their associated concepts. Counts for the binarized version of the concepts provided where applicable to illustrate the problem size.

Boxing, as implemented in OCAtari (Delfosse et al., 2023), uses reverse engineering to extract positions of important objects from the game's RAM state. This extraction process introduces a small amount of noise. The concepts in Pong are derived from the VIPER paper (Bastani et al., 2018) that also uses reverse engineering.

Baselines. To our knowledge, there exist no previous algorithms that seek to minimize the number of concept labels for interpretable RL, so we implement three: Sequential-Q, Disagreement-Q, and Random-Q. In Sequential-Q, the agent spends B queries on the first B states encountered during the initial policy rollout. In Disagreement-Q, the agent also spends its budget at the beginning of training; however, it uses active learning with the same $\alpha(\cdot)$ as LICORICE to strategically choose the training data for annotation. In Random-Q, the agent receives B concept labels at random points using a probability to decide whether to query in each state. As a budget-unconstrained baseline, we implement CPM from previous work in multi-agent RL (Zabounidis et al., 2023) for the single-agent setting. As mentioned in §2, CPM jointly trains g and f, assuming unlimited access to concept labels, so it also represents an upper bound on concept accuracy.

VLM Details. For our VLM experiments, we use GPT-4o (gpt). During training, we query GPT-4o each time LICORICE requires a concept label. As an example, for the concept related to the position of an object, we prompt GPT-4o with instructions to report the coordinates (x y). More details about our prompts can be found in Appendix A.3.

Performance Metrics. We present the reward as a function of the upper bound calculated by training PPO with ground-truth concept labels (PixelCartPole: 500, DoorKey: 0.97, DynamicObstacles: 0.91, Boxing: 86.05, Pong: 21). In the first four environments, percentages (or ratios) make sense since the minimum reasonable reward is 0.[2] In Pong, a random policy has -21 reward so we first get the difference between rewards and -21 and then calculate the ratio. We additionally report the concept error (MSE for regression; 1 - accuracy for classification). In Boxing, following the practice of OCAtari, we consider a concept prediction correct if it is within 5 pixels of the ground truth label. In Pong, we simply use classification error. All reported numbers are calculated using 100 evaluation episode s.

4.2 RESULTS

RQ1: REWARD AND CONCEPT ACCURACY UNDER A LIMITED ANNOTATION BUDGET

LICORICE achieves similarly high reward and concept performance to the state-of-the-art budget-unconstrained baseline. We first seek to understand how LICORICE performs compared to the state-of-the-art in concept-based RL, CPM, which is not constrained by concept label budgets. Surprisingly, LICORICE and CPM achieve nearly 100% of the maximum reward in all environments in this unfair comparison (Figure 2). They also achieve a similar concept error rate, only seeing a clear difference in error for the most complex environment, Pong. We emphasize that CPM is given an unlimited budget; in practice, it uses 1M-15M concept labels for all environments, which is around 2000 to 13000× the budget used by LICORICE. In contrast, LICORICE operates under a strict budget constraint, with 3000 labels for Boxing and 5000 for Pong, and at most 500 concept labels for the simpler environments (PixelCartPole: 500, DoorKey: 300, DynamicObstacles: 300).

---

[2] In PixelCartPole and DoorKey, all rewards are nonnegative; in DynamicObstacles, an agent can ensure nonnegative reward by simply staying in place; in Boxing, a random policy can get around 0 reward and all of the trained policies have positive reward.

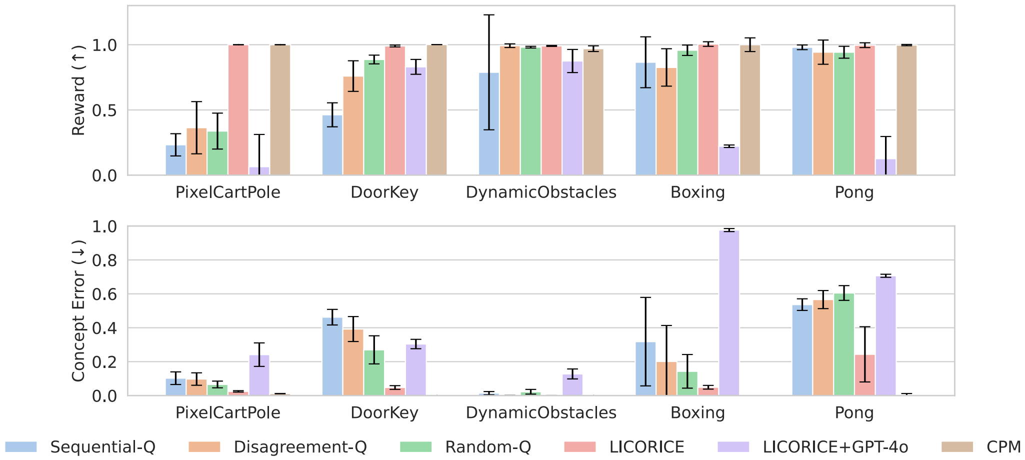

Figure 2: LICORICE generally achieves higher reward and lower concept error than all other budget-constrained algorithms (Sequential-Q, Disagreement-Q, Random-Q). The budgets for each environment are PixelCartpole: 500, DoorKey: 300, DynamicObstacles: 300, Boxing: 3000, Pong: 5000. CPM has an unlimited budget (in practice, it uses 4M, 4M, 1M, 15M, 10M labels, respectively). Despite this extreme budget difference, LICORICE achieves comparable reward and concept error. Full results in Table 5, Appendix B.

LICORICE generally achieves higher reward and lower concept error than budget-constrained baselines. We now study all budget-constrained algorithms (LICORICE plus those in §4.1) under the same fixed concept labeling budget as above. Figure 2 shows that LICORICE outperforms all baselines in terms of reward and concept error on all but arguably the easiest environment, DynamicObstacles. In that environment, LICORICE performs similarly to the baselines. The greatest performance differences occur in PixelCartPole and DoorKey, where LICORICE achieves 100% and 99% of the maximum reward, while the second-best algorithm achieves 36% and 89%, respectively. We therefore answer **RQ 1** affirmatively: not only does LICORICE achieve the same high concept accuracy and high reward as the state-of-the-art in concept-based RL, it also performs on-par to budget-constrained baselines in one environment and outperforms them in the other four.

RQ 2: VLMS AS CONCEPT ANNOTATORS

We now seek to answer the question of whether VLMs can successfully provide concept labels in lieu of a human annotator within our LICORICE framework. Because using VLMs incurs costs and users requiring interpretable policies for their environments may still face budget constraints, we operate within the same budget-constrained setting as above. We present these results in Figure 2.

VLMs can serve as concept annotators for some environments. In DoorKey and DynamicObstacles, LICORICE with GPT-4o labels achieves 83% and 88% of the maximum reward, respectively. To assess the quality of VLM-generated labels, we compare the concept error rate of our trained model against GPT-4o's labeling error, both evaluated on the same rollout observations, while GPT-4o evaluation is on a random subset of 50 observations, due to API cost constraints. Table 2 shows that the concept error of LICORICE is comparable to that of GPT-4o when GPT-4o labeling error is not high, which suggests that LICORICE can effectively utilize these concept labels. VLMs, particularly GPT-4o, could serve as concept annotators for certain concepts in certain environments.

	c Error	
Environment	LICORICE+GPT-4o	GPT-4o
PixelCartPole	0.25	0.24
DoorKey	0.31	0.30
DynamicObstacles	0.13	0.13
Boxing	0.98	0.82
Pong	0.73	0.87

Table 2: Concept error comparison. The concept error of LICORICE+GPT-4o matches that of GPT-4o alone.

VLMs struggle to provide accurate concepts in more complex environments. In PixelCartPole, Boxing, and Pong, however, GPT-4o struggles to provide accurate labels. In these environments, not

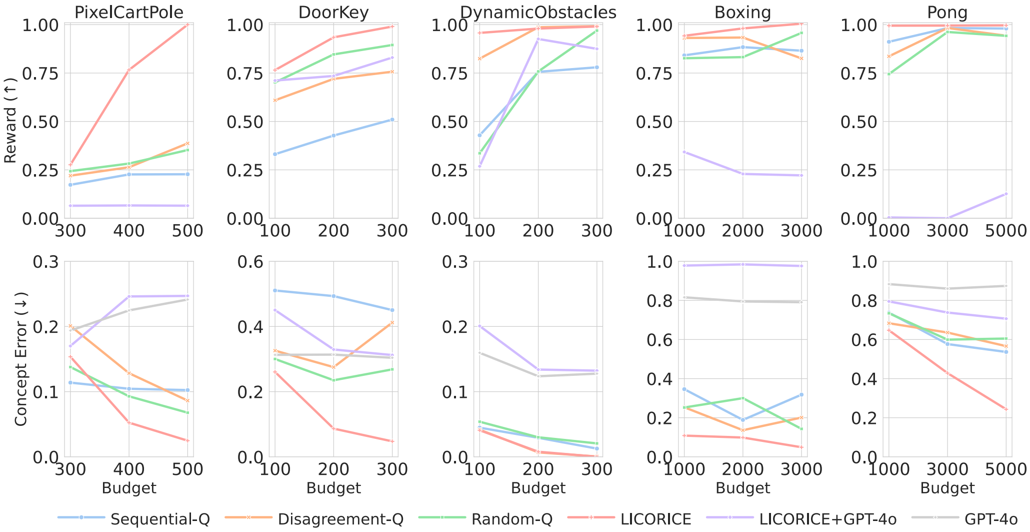

Figure 3: Reward (top) and concept error (bottom) of all algorithms for different budgets. LICORICE more efficiently makes use of the varying budgets, achieving higher reward and lower concept error. Full results with standard deviation are in Tables 6 to 8, Appendix B.

only does LICORICE+GPT-4o achieve less than 25% of the maximum reward, it also incurs a large concept error. The challenge with PixelCartPole is the continuous nature of the concepts and the lack of necessary knowledge of physical rules and quantities in this specific environment. Boxing is particularly challenging due to the large number of possible concept values that each of the 8 discrete concepts can take on (160 or 210 possible values). Similarly, concepts in Pong have hundreds of possible values and some physics quantities require multi-frame inference. We therefore answer **RQ 2** with cautious optimism: there are indeed cases in which LICORICE could be used alongside VLMs. Generally speaking, a less complex environment is more likely to work well with VLMs by enabling clear and detailed labeling instructions in the prompt. However, VLM capabilities need improvement before they can fully alleviate the human annotation burden in safety-critical settings.

RQ 3: INVESTIGATION OF BUDGET REQUIREMENTS FOR PERFORMANCE

Given these results, we now investigate the minimum budget required for all environments to achieve 99% of the reward upper bound. As a result, we incrementally increase the budget for each environment starting from a reasonably small budget until LICORICE reaches 99% of the reward upper bound. In Pong, we further increase the budget until the concept error is below 0.25. In addition to the baselines, we study LICORICE under perfect (human) annotation and noisy VLM labels in Figure 3.

LICORICE more rapidly achieves high reward compared to baselines. Across all environments, LICORICE is the most query-efficient. In three of the environments, it consistently achieves higher reward than baselines across all budget levels. In DynamicObstacles and Pong, LICORICE outperforms baselines at the smallest query budget, after which some baseline method achieves comparable reward. LICORICE is similarly performant with respect to concept error. Except for one budget setting for one environment, LICORICE consistently achieves lower concept error than the baselines.

For LICORICE+GPT-4o, more concept labels is not always better. In DoorKey, LICORICE+GPT-4o exhibits predictable behavior in terms of concept error: as the budget increases, the reward increases and the concept error decreases. Counterintuitively, for some environments, the concept error and reward fluctuates with more budget, likely due to the additional labeling noise introduced to the concept network. We therefore provide a more nuanced answer to **RQ 3**: LICORICE is indeed label efficient across varying budgets, but the benefit is annotator-dependent.

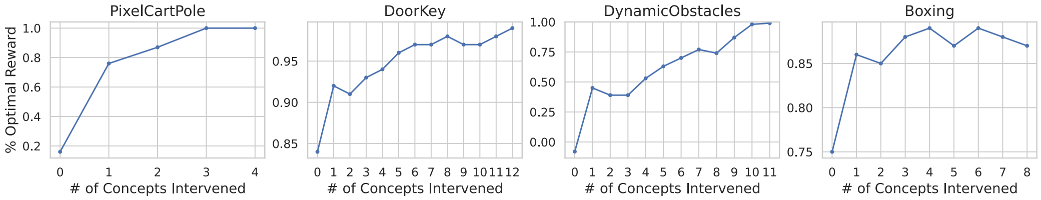

Figure 4: Concept intervention results: LICORICE enables test-time concept intervention.

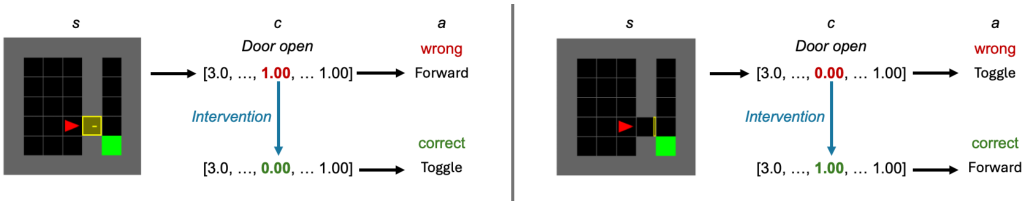

Figure 5: Test-time intervention examples, where intervening on a single concept corrects the action.

RQ 4: INTERPRETABILITY ANALYSIS: TEST-TIME CONCEPT INTERVENTIONS

A great property of concept-based networks is the ability for people to successfully intervene on the concepts to correct them. In RL, this intervention enables the inspection of how different concepts influence the immediate decisions of the agent.

LICORICE enables test-time concept intervention. To validate that LICORICE supports test-time concept intervention, we simulate using a noisy concept model g with f to study the impact of concept interventions on reward. For PixelCartPole, each concept adds a Gaussian noise with standard deviation of 0.2. For the other three environments with discrete concepts, each concept label is randomly changed with probability 0.2. Following previous work (Koh et al., 2020), we first intervene individually on concepts, setting them to ground truth, and sort the concepts in descending order of reward improvement. We then sequentially intervene on the concepts following this ordering, such that any previously intervened concepts remain intervened. According to Figure 4, using a noisy concept model significantly degrades reward. However, the performance increases with more interventions, meaning LICORICE supports test-time concept intervention, affirmatively answering **RQ 4**.

Concept interventions show how the policy decisions change. We now show how domain experts could interact with the model. Specifically, we simulate a counterfactual question: What if a concept value is incorrectly predicted? Figure 5 depicts two examples in DoorKey. Just intervening on the concept of the door being open or closed is sufficient to change the agent's behavior, both highlighting the importance of this specific concept and the ability of a user to intervene on the concepts to interrogate and understand agent behavior.

ADDITIONAL EXPERIMENTS

Ablation study: all components of LICORICE contribute positively to its performance. We now conduct an ablation study of our three main contributions: iterative training, decorrelation, and disagreement-based active learning. LICORICE-IT uses only one iteration, LICORICE-DE does not perform data decorrelation, and LICORICE-AC uses the entire unlabeled dataset for querying instead of a subset chosen by active learning. All components are critical to achieving both high reward and low concept error (Table 3; learning curves in Appendix B). However, the component that most contributes to the performance differs depending on the environment. For example, compared with LICORICE, LICORICE-IT exhibits the largest reward gap for PixelCartPole; however, LICORICE-AC yields the largest reward gap for DynamicObstacles, and LICORICE-DE yields the largest gap for DoorKey. This difference may occur because the concepts in DynamicObstacles are simple enough that one iteration is sufficient, so the largest gains can be made with active learning. In contrast, PixelCartPole requires further policy refinement to better estimate on-distribution concept values, so the largest gains can be made by leveraging multiple iterations.

	PixelCartPole		DoorKey		DynamicObstacles		Boxing	
Algorithm	$R\uparrow$	c MSE $\downarrow$	$R\uparrow$	c Error $\downarrow$	$R\uparrow$	c Error $\downarrow$	$R\uparrow$	c Error $\downarrow$
LICORICE-IT	0.53	0.08	0.99	0.09	1.00	0.00	0.98	0.14
LICORICE-DE	0.97	0.03	0.78	0.20	0.98	0.00	1.00	0.04
LICORICE-AC	0.91	0.02	0.82	0.16	0.58	0.00	0.99	0.09
LICORICE	1.00	0.02	0.99	0.05	0.99	0.00	1.00	0.05

Table 3: Ablation study results for LICORICE. All components generally help achieve better reward and lower concept error. Full results with standard deviation are in Table 9, Appendix B.

LICORICE is robust to various hyperparameter values. LICORICE involves a few key hyperparameters, described in §3 and summarized here for convenience. In data decorrelation, τ governs the size of the unlabeled dataset collected by the current policy and p controls the rate of acceptance of data points. Active learning involves a concept ensemble with N models. We study the effect of the values of these hyperparameters on DynamicObstacles, finding that LICORICE achieves $96 - 99\%$ of the maximum reward across 2 values for N, 3 values for p, and 3 values for τ (Appendix B.5).

5 RELATED WORK

Interpretable RL. Interpretable RL has gained significant attention in recent years (Glanois et al., 2024). One prominent area uses rule-based methods—such as decision trees (Silva et al., 2020; Topin et al., 2021), logic (Delfosse et al., 2024), and programs (Verma et al., 2018; Penkov & Ramamoorthy, 2019)—to represent policies. These works either assume that the state is already interpretable or that the policy is pre-specified. Unlike prior work, our method involves *learning* the interpretable representation (through concept training) for policy learning.

Concept Learning for RL. Due to successes in the supervised setting (Collins et al., 2023; Sheth & Ebrahimi Kahou, 2023; Zarlenga et al., 2023), concept models have recently been used in RL. Das et al. (2023) trains a joint embedding model between state-action pairs and concept-based explanations to expedite learning via reward shaping. Unlike LICORICE, their policy is not a strict function of the concepts, allowing our techniques to be combined to provide concept-based explanations alongside an interpretable policy. Another example, CPM (Zabounidis et al., 2023), is a multi-agent RL approach that assumes that concept labels are continuously available during training. As we have shown, this approach uses over 1M concept labels, whereas LICORICE achieves similar performance while using 2000× fewer labels.

Learning Concepts with Human Feedback. Prior research has explored leveraging human concept labels, but not for RL and without focusing on reducing the labeling burden. For instance, Lage & Doshi-Velez (2020) has users label additional information about the relevance of certain feature dimensions to the concept label to facilitate concept learning for high-dimensional data. In a different vein, Chauhan et al. (2023) develop a test-time intervention policy for querying for concept labels to improve the final prediction. These methods do not directly tackle the issue of reducing the overall labeling burden during training and could be used alongside our method. A concurrent work (Shao et al., 2024) also studies sample efficiency but mainly focus on human task demonstrations.

6 DISCUSSION AND CONCLUSION

In this work, we proposed LICORICE, a novel RL algorithm that addresses the critical issue of model interpretability while minimizing the reliance on continuous human annotation. We introduced a training scheme that enables RL algorithms to learn concepts efficiently from little labeled concept data. We demonstrated how this approach reduces manual labeling effort. We conducted initial experiments to demonstrate how we can leverage powerful VLMs to infer concepts from raw visual inputs without explicit labels in some environments. There are broader societal impacts of our work that must be considered, including both the impacts of using VLMs in real-world applications and more general considerations around interpretability. For a more detailed discussion of limitations and future work, please refer to Appendix C.

REFERENCES

Hello gpt-4o. https://openai.com/index/hello-gpt-4o/.

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

O Bastani, Y Pu, and A Solar-Lezama. Verifiable reinforcement learning via policy extraction. *Advances in Neural Information Processing Systems*, 2018.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

William H. Beluch, Tim Genewein, A. Nürnberger, and Jan M. Köhler. The power of ensembles for active learning in image classification. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9368–9377, 2018. URL https://api.semanticscholar.org/CorpusID:52838058.

Kushal Chauhan, Rishabh Tiwari, Jan Freyberg, Pradeep Shenoy, and Krishnamurthy Dvijotham. Interactive concept bottleneck models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 5948–5955, 2023.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

Katherine Maeve Collins, Matthew Barker, Mateo Espinosa Zarlenga, Naveen Raman, Umang Bhatt, Mateja Jamnik, Ilia Sucholutsky, Adrian Weller, and Krishnamurthy Dvijotham. Human uncertainty in concept-based ai systems. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 869–889, 2023.

Devleena Das, Sonia Chernova, and Been Kim. State2explanation: Concept-based explanations to benefit agent learning and user understanding. *Advances in Neural Information Processing Systems*, 36, 2023.

Quentin Delfosse, Jannis Blüml, Bjarne Gregori, Sebastian Sztwiertnia, and Kristian Kersting. Ocatari: Object-centric atari 2600 reinforcement learning environments. 2023.

Quentin Delfosse, Hikaru Shindo, Devendra Dhami, and Kristian Kersting. Interpretable and explainable logical policies via neurally guided symbolic abstraction. *Advances in Neural Information Processing Systems*, 36, 2024.

Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, et al. Concept embedding models: Beyond the accuracy-explainability trade-off. *Advances in Neural Information Processing Systems*, 35:21400–21413, 2022.

Claire Glanois, Paul Weng, Matthieu Zimmer, Dong Li, Tianpei Yang, Jianye Hao, and Wulong Liu. A survey on interpretable reinforcement learning. *Machine Learning*, pp. 1–44, 2024.

Niko Grupen, Natasha Jaques, Been Kim, and Shayegan Omidshafiei. Concept-based understanding of emergent multi-agent behavior. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

Harmanpreet Kaur, Harsha Nori, Samuel Jenkins, Rich Caruana, Hanna Wallach, and Jennifer Wortman Vaughan. Interpreting interpretability: understanding data scientists' use of interpretability tools for machine learning. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pp. 1–14, 2020.

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

Isaac Lage and Finale Doshi-Velez. Learning interpretable concept-based models with human feedback. *International Conference on Machine Learning: Workshop on Human Interpretability in Machine Learning*, 2020.

Xiao-Yang Liu, Ziyi Xia, Jingyang Rui, Jiechao Gao, Hongyang Yang, Ming Zhu, Christina Wang, Zhaoran Wang, and Jian Guo. Finrl-meta: Market environments and benchmarks for data-driven financial reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1835–1849, 2022.

Jiayuan Mao, Tomás Lozano-Pérez, Josh Tenenbaum, and Leslie Kaelbling. Pdsketch: Integrated domain programming, learning, and planning. *Advances in Neural Information Processing Systems*, 35:36972–36984, 2022.

Catherine C Marshall and Frank M Shipman. Experiences surveying the crowd: Reflections on methods, participation, and reliability. In *Proceedings of the 5th Annual ACM Web Science Conference*, pp. 234–243, 2013.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

Hussein Mozannar, Hunter Lang, Dennis Wei, Prasanna Sattigeri, Subhro Das, and David Sontag. Who should predict? exact algorithms for learning to defer to humans. In *AISTATS*, 2023.

Tuomas Oikarinen, Subhro Das, Lam M Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. *arXiv preprint arXiv:2304.06129*, 2023.

Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Svetlin Penkov and Subramanian Ramamoorthy. Learning programmatically structured representations with perceptor gradients. In *Proceedings of the International Conference on Learning Representations*, 2019.

Eleonora Poeta, Gabriele Ciravegna, Eliana Pastor, Tania Cerquitelli, and Elena Baralis. Concept-based explainable artificial intelligence: A survey. *arXiv preprint arXiv:2312.12936*, 2023.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

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

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 287–294, 1992.

Jie-Jing Shao, Hao-Ran Hao, Xiao-Wen Yang, and Yu-Feng Li. Learning for long-horizon planning via neuro-symbolic abductive imitation, 2024. URL `https://arxiv.org/abs/2411.18201`.

Ivaxi Sheth and Samira Ebrahimi Kahou. Auxiliary losses for learning generalizable concept-based models. *Advances in Neural Information Processing Systems*, 36, 2023.

Andrew Silva, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *International conference on artificial intelligence and statistics*, pp. 1855–1865. PMLR, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Alaa Tharwat and Wolfram Schenck. A survey on active learning: state-of-the-art, practical challenges and research directions. *Mathematics*, 11(4):820, 2023.

Nicholay Topin, Stephanie Milani, Fei Fang, and Manuela Veloso. Iterative bounding mdps: Learning interpretable policies via non-interpretable methods. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9923–9931, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 10, 2017.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pp. 5045–5054. PMLR, 2018.

Renhao Wang, Jiayuan Mao, Joy Hsu, Hang Zhao, Jiajun Wu, and Yang Gao. Programmatically grounded, compositionally generalizable robotic manipulation. *International Conference on Learning Representations*, 2023.

Chao-Han Huck Yang, I Danny, Te Hung, Yi Ouyang, and Pin-Yu Chen. Causal inference q-network: Toward resilient reinforcement learning. In *Self-Supervision for Reinforcement Learning Workshop-ICLR 2021*, 2021.

Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021.

Renos Zabounidis, Joseph Campbell, Simon Stepputtis, Dana Hughes, and Katia P Sycara. Concept learning for interpretable multi-agent reinforcement learning. In *Conference on Robot Learning*, pp. 1828–1837. PMLR, 2023.

Mateo Espinosa Zarlenga, Katherine M Collins, Krishnamurthy Dvijotham, Adrian Weller, Zohreh Shams, and Mateja Jamnik. Learning to receive help: Intervention-aware concept embedding models. *Advances in Neural Information Processing Systems*, 2023.