Concept-Based Interpretable Reinforcement Learning with Limited to No Human Labels

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-based 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 neural networks. However, a significant limitation in prior work is the assumption that human annotations for these concepts are readily available during training, necessitating continuous real-time input from human annotators. To overcome this limitation, we introduce a novel training scheme that enables RL algorithms to efficiently learn a concept-based policy by only querying humans to label a small set of data, or in the extreme case, without any human labels. Our algorithm, LICORICE, involves three main contributions: interleaving concept learning and RL training, using a concept ensembles to actively select informative data points for labeling, and decorrelating the concept data with a simple strategy. We show how LICORICE reduces manual labeling efforts to to 500 or fewer concept labels in three environments. Finally, we present an initial study to explore how we can use powerful vision-language models to infer concepts from raw visual inputs without explicit labels at minimal cost to performance.

1 Introduction

In reinforcement learning (RL), agents are tasked with learning a *policy*, a rule that makes sequential, reactive decisions in complex environments. In recent RL work, agents typically represent the policy as a neural network, as such representations tend to lead to high performance [17]. However, this choice can come at a cost: such policies are challenging for stakeholders to interpret — particularly when the inputs to the network are also complex, such as high-dimensional sensor data. This opacity can become a significant hurdle, especially in applications where understanding the rationale behind decisions is critical, such as healthcare [32] or finance [15]. In such applications, decisions can have significant consequences, so it is essential for stakeholders to fully grasp the reasoning behind actions in order to confidently adopt or collaborate on a policy.

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

Preprint. Under review.

To address interpretability concerns in the supervised learning setting, recent works have integrated human-understandable concepts into the decision-making process through concept bottleneck models [13, 9]. These models incorporate a bottleneck layer whose units correspond to interpretable concepts, ensuring that the final decisions depend on these concepts instead of just on opaque raw inputs. By training the model both to have high accuracy on the target task 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.

More recently, these techniques have been applied to RL by incorporating a concept bottleneck in the policy [11, 33], such that the actions taken by the agent are a function of the human-understandable concepts. However, past work assumes that human-provided concept annotations are accessible in the *training* loop of RL, which presents a significant challenge. In novel domains, we may only have access to data that is not in an interpretable representation, such as RGB or multispectral satellite imagery, while the RL agent needs to know the concepts present in every state and action it encounters during training — even though RL training sets can often be measured in millions or billions of state-action pairs. As a result, the human labelers would need to be actively involved in the RL training process, providing concept labels for unreasonable numbers of observations. This level of involvement is not only detrimental to the human labeler but also risks introducing potential biases and errors in the model training process due to fatigue [16].

In this work, we aim to address this challenge of requiring frequent human interventions during the training of interpretable policies using RL. We propose LICORICE (**L**abel-efficient **I**nterpretable **CO**ncept-based **R**e**I**nfor**CE**ment learning), a novel training paradigm consisting of three main contributions. First, LICORICE interleaves concept learning and RL training: it alternately freezes the network layers corresponding to either the concept learning part or the decision-making part. We believe this scheme improves our ability to learn from limited labeled data by reducing interference between the learning tasks. Additionally, concept learning uses data that is more on-policy, which means it is more relevant and useful for learning accurate concepts that directly impact the decision-making process. Second, LICORICE utilizes concept ensembles to actively select the most informative data points for labeling. By focusing on samples that are predicted to provide the most valuable information for model improvement, this technique substantially reduces the number of labels needed to achieve both high performance and high concept accuracy. Third, LICORICE includes a strategy to decorrelate the concept data collected under the current policy. By generating a diverse set of training data, this approach ensures the data remains unbiased and representative of various scenarios. We demonstrate how these changes yield both higher concept accuracy and higher reward while requiring fewer queries on three environments with image input, including an image input version of CartPole and two Minigrid environments.

Given these results, we ask whether we can further reduce the reliance on manual concept labeling by leveraging the potential of vision-language models (VLMs). This capability is important in scenarios where manual concept annotation is impractical. We present an initial exploration of whether VLMs can further reduce the concept annotation burden. We find that VLMs can indeed serve as concept annotators for some but not all of the above environments. In these cases, the resulting policies can achieve 83 to 93% of the optimal performance.

2 Preliminaries

2.1 Reinforcement Learning

In RL, an agent learns to make decisions by interacting with an environment [27]. The environment is commonly modeled as a Markov decision process [21], 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 actions with the aim of maximizing the expected cumulative discounted reward. We evaluate a policy via its value function, which is defined as $V^\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_0 = s], \forall s \in \mathcal{S}$. The ultimate aim in RL is to determine the optimal policy, π^*. To do so, the agent iteratively refines its policy based on feedback from the environment.

2.2 Concept Bottleneck Models

Concept-based explanations have emerged as a common paradigm in explainable AI [20]. They explain a model's decision-making process through human-interpretable attributes and abstractions. Concept bottleneck models [13] are an example of concept-based explanations: they learn a mapping from samples $x \in X$ to labels $y \in Y$ through two functions, g and f. The concept encoder function $g : X \to C$ maps samples from the input space to an intermediate space of human-interpretable concepts. The label predictor function $f : C \to Y$ maps samples from the concept space to a downstream task space, such as labels for supervised classification. As a result, the prediction $\hat{y} = f(g(x))$ relies on the input x entirely through the bottleneck $\hat{c} = g(x)$.

Training these models requires a dataset of $X \times C \times Y$, in which each sample consists of input features x, a ground truth concept vector $c \in \{0, 1\}^k$, and a task label y. The functions f and g are parameterized by neural networks. These models are trained independently, jointly, or sequentially. The *independent* training method trains $\hat{f}$ and $\hat{g}$ independently. Critically, $\hat{f}$ is trained using the true c, but at test time it takes $\hat{g}(x)$ as input. If directly applied to the RL setting, this approach would require continuous access to ground-truth concepts for training $\hat{f}$, which we would like to avoid. In *joint* training, the model minimizes the weighted sum of the concept losses with the task performance loss. This paradigm can be problematic, as the losses can interfere with each other, and it may require careful tuning to find the right balance between the two losses. In *sequential* training, the model first learns $\hat{g}$. It then uses the concept predictions $\hat{g}(x)$ to learn $\hat{f}(\hat{g}(x))$. As we will later show, this setup is problematic for RL, as concepts may only emerge after the policy is sufficiently performant.

3 LICORICE

We now describe LICORICE, our algorithm for interactively querying for concept labels under a limited concept label budget during RL training. In Section 3.1, we describe the architecture of the concept bottleneck policy. We then detail our iterative training process (Section 3.2) and explain our active learning approach for choosing informative training points (Section 3.3). Finally, in Section 3.4, we detail how we use VLMs as a substitute for human concept labeling.

3.1 Concept Bottleneck Policies

We insert a concept bottleneck layer into the policy network, such that π maps from states $\mathcal{S}$ to concepts C to actions $\mathcal{A}$, $\pi = f(g(s))$. These concepts serve as intermediaries in the policy network, which subsequently maps the concept vector to a distribution over possible actions. This setup allows the policy to base its decisions on understandable and meaningful features. We describe additional modifications necessary to accommodate the concept bottleneck in Appendix A.2. As a result, we can use any RL algorithm as long as we include an additional loss function for concept prediction.

Loss Function We now describe how we use two loss functions for training in our iterative training scheme, described in more detail in Section 3.2. Because we employ an *iterative* training scheme, we disentangle the two loss functions — the concept prediction loss and the standard RL loss — to prevent interference between them. In this way, the concept prediction loss L^C only affects g, the part of the model responsible for predicting concepts, and the standard RL loss L^{RL} only affects f. For training g, we employ two types of loss functions depending on the nature of the concepts (MSE for continous concepts; cross-entropy loss for categorical concepts). If the problem requires mixed-type concepts, we could discretize the continuous attributes, converting them into categorical forms suitable for classification. This approach ensures that our method accommodates a diverse range of concept types.

3.2 Training Process

We now present LICORICE, our novel algorithm for optimizing concept-based models for RL within a fixed concept query budget. Our iterative training approach extends the sequential bottleneck training paradigm by incorporating multiple phases. Intuitively, this strategy can be advantageous for obtaining accurate on-policy concept estimates. To collect candidate training data for g, we employ our current policy for rollouts. To minimize data point correlation and collect a diverse range of data,

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

1: **Input:** Total budget B, number of iterations M, sample acceptance threshold p, ratio τ 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 π_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: $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{train}} \cup \mathcal{D}'_{\text{train}}$, $\mathcal{D}_{\text{val}} \leftarrow \mathcal{D}_{\text{val}} \cup \mathcal{D}'_{\text{validation}}$
18:     Continue training the concept network g on $\mathcal{D}_{\text{train}}$, using $\mathcal{D}_{\text{val}}$ for early stopping
19:     Freeze g and train f using standard RL training to obtain π_{m+1}
20: **end for**

we use a decorrelation strategy. To select more informative points to query for concept labels, we use a concept ensemble to calculate disagreement.

More specifically, LICORICE (Algorithm 1) proceeds in three main stages within each iteration $m \in M$: data collection, concept ensemble training for data selection, and concept bottleneck policy training. First, in the data collection stage (lines 4 to 8), we collect a dataset of unlabeled concept data $\mathcal{U}_m$ by rolling out our current policy π_m. Crucially, during this step, we decorrelate the data to ensure the data points are diverse (line 6). Specifically, we set an acceptance probability p for adding the data point to the dataset, as they are generated by the temporally-correlated RL policy. This prepares a diverse dataset for the next stage, with the aim of promoting the training of a more performant concept ensemble for data selection and, eventually, a more accurate concept bottleneck portion of the policy.

The second stage is concept ensemble training for data selection (lines 9 to 16). In this stage, we train the 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, which evaluates whether each candidate in our unlabeled dataset $\mathcal{U}_m$ ought to receive a ground-truth concept label (line 11). This function targets samples where prediction disagreement is highest, detailed in Section 3.3. We then query for B_m ground-truth concept labels (line 13) to prepare us for the next stage.

We are now prepared for the third stage: concept bottleneck policy training (lines 17 to 19). We aggregate the concept datasets from the second stage with data from previous iterations, and continue training the concept network g on $\mathcal{D}_{\text{train}}$ (line 18). After early stopping dictates that we stop g training, we freeze g and train f using any standard RL algorithm — for example, PPO — to obtain the policy for collecting data in the next iteration (line 19). With that, we are prepared to start the process again from the first stage.

3.3 Active Concept Learning

We propose to leverage an ensemble of concept models to calculate the disagreement-based acquisition function for actively querying for concept labels. To quantify the concept disagreement of a state $U(s)$, we use two different formulations, depending on the concept learning task.

Classification For concept classification, we use a query-by-committee [24] approach. When the models produce diverse predictions, it indicates that the instance is difficult and more information would be particularly valuable. Conversely, if all models agree, the instance is likely already well understood, and additional labels would be less beneficial. More specifically, we prioritize points with a high proportion of predicted class labels that are not the modal class prediction (also called the variation ratio [3]). This is given by $U(s) = 1 - \max_{c \in C} \frac{1}{N} \sum_{i=1}^{N} [\tilde{g}_i(s) = c]$, where $\tilde{g}_i(s)$ is the concept prediction of the i-th model on state s.

Regression For concept regression, we observe that we can instead directly use variance as a measure of disagreement. Specifically, the concept disagreement of a state is quantified by the unbiased estimation of variance $U(s) = \sigma^2(s)$ of the predictions across the concept models, due to Bessel's correction, defined as: $U(s) = \sigma^2(s) = \frac{1}{N-1} \sum_{i=1}^{N} (\tilde{g}_i(s) - \mu(s))^2$, where $\mu(s) = \frac{1}{N} \sum_{i=1}^{N} \tilde{g}_i(s)$ is the mean prediction of the N concept models.

3.4 Using Vision-Language Models for Concept Labeling

Equipped with our overall algorithm, we now seek to further reduce human annotation burden. To do so, we turn to VLMs due to their remarkable performance in understanding and generating human-like descriptions of visual content [22]. Within the pipeline of LICORICE, we keep all aspects of our algorithm the same but use the VLM as the concept annotator in the training loop. By doing so, we effectively decrease the human annotation effort to zero, albeit with some labeling inaccuracy introduced.

In our experiments, we use GPT-4o [1], a closed-source model that is possibly the most capable vision-language model in the world at the time of writing the paper. During the training loop of LICORICE, we query GPT-4o each time the algorithm requires a concept label (line 13 in Algorithm 1). As we are using a pre-specified concept set, we design prompts with detailed general descriptions of the scene layout and definitions of all concepts, in a similar fashion to giving labeling instructions to real humans. We then prompt the GPT-4o to obtain the generated labels. In environments where continuous concept values are required, we ask GPT-4o to give as accurate estimates as possible; in environments with concepts that are discrete and more intuitive to label, we provide clear instructions of how to read the concept numbers according to the input image. More details about our prompts can be found in Appendix A.3.

4 Experiments

In our experiments, we investigate the following questions:

RQ 1 Does LICORICE enable both high concept accuracy and high environment reward?

RQ 2 To what extent does LICORICE enable a reduction in human labeling effort without substantially impacting performance?

RQ 3 Can LICORICE be used alongside powerful vision-language models to further alleviate the labeling burden?

4.1 Environments

We evaluate our approach on three environments. For each environment, we define an interpretable concept set, where we define concepts to describe properties of objects of interest.

PixelCartPole In PixelCartPole-v1 [31], the states are images, and the concepts are the original continuous features in the standard CartPole environment: the cart position, the cart velocity, the pole angle, and the pole angular velocity. To obtain the mapping from images to concepts, we use a fixed window of the four most recent images for the temporal concepts, such as the cart velocity. We also include the last action as input to ensure that the information is sufficient to infer concept values. This domain is deceptively difficult due to the temporal concepts.

DoorKey In DoorKey-7x7 [5], the agent operates in a 5x5 grid[3] to pick up an item to unlock the door, then reach the green goal square. The states are fully observable images. We define 6 groups of concepts with 12 concept variables in total: 1) the agent's position, 2) the agent's direction, 3) the key's position or a fixed position (0, 0) outside the grid if the key has been picked up, 4) the door's position, 5) whether the door is open, and 6) whether the agent can move along each of four directions, i.e., the corresponding cell is empty. Each group contains one or more concept variables. For example, position groups contain x and y coordinates.

DynamicObstacles In DynamicObstacles-5x5 [5], the agent operates in a 3x3 grid to avoid two moving obstacles while navigating to the goal. Colliding with the dynamic obstacles yields a large penalty, so the agent must correctly learn concepts to safely reach the goal. The states are fully observable images. We define 5 groups of concepts with 11 concept variables in total: 1) the agent's position, 2) the agent's direction, 3) the first obstacle's position, 4) the second obstacle's position, and 5) whether the agent can move along each of four directions, i.e., the corresponding cell is empty. Each group contains one or more concept variables. For example, position groups contain x and y coordinates.

4.2 Experiment Details

In each experiment, we run each algorithm 5 times, each with a random seed. More implementation details and hyperparameters are in Appendix A.2.

Architecture For the backbone model and algorithm, we use PPO (implementation provided by Stable Baselines 3 [23]). We preserve most of the network architecture, but we add a concept layer in the policy network. Specifically, after passing the image through the feature extractor, which consists of 3 CNN layers, we introduce a linear layer to map to the concept layer and obtain predicted concept values. Each predicted concept is represented as a real value for the regression case and one categorical value derived from a classification head for the classification case. This portion of the network corresponds to g. For f, we utilize an MLP extractor with 2 fully connected layers, each with 64 neurons and a Tanh activation function. We share the default CNN feature extractor between policy and value functions, as we found this to be beneficial in preliminary experiments.

Concept Representation We model concept learning for PixelCartPole-v1 as a regression problem (minimizing mean squared error) as the concepts are real-valued, and concept learning for DoorKey-7x7 and DynamicObstacles-5x5 as classification problems as the concepts are categorical. The observation images have resolutions of $(240, 160)$ for PixelCartPole-v1, and $(160, 160)$ for DoorKey-7x7 and DynamicObstacles-5x5. These dimensions maintain a short side of 160 pixels while preserving the aspect ratio of the original rendered images, as we observe lower resolutions may impact performance.

Performance Metrics To obtain an upper bound on the reward for each environment, we directly use ground-truth concept labels to learn a policy, leading to a reward of 500, 0.97, and 0.91 for the three environments respectively. In the following sections, we report the reward as a ratio of this upper bound. Percentages (or ratios) make sense here since the minimum reasonable reward is 0 in all environments: in PixelCartPole-v1 and DoorKey-7x7, all rewards are nonnegative; in DynamicObstacles-5x5, a random policy would have negative reward due to collisions, but the agent can ensure nonnegative reward by simply staying in place. We additionally report the concept error (MSE for regression; 1 - accuracy for classification). All the reported numbers are calculated during the testing stage, where we evaluate the models on 100 episodes.

4.3 Results

4.3.1 Balancing Concept Performance and Environment Reward

We first validate that LICORICE can achieve high reward in all test environments while accurately identifying concepts. We first compare against baselines with a *fixed* budget of $B = [500, 300, 300]$

[3]In both of the Minigrid environments, the actual usable area is smaller than what shows in its name, as the outermost layer is a boundary.

c Labels	Algorithm	PixelCartPole-v1		DoorKey-7x7		DynamicObstacles-5x5	
		$R \uparrow$	c MSE $\downarrow$	$R \uparrow$	c Error $\downarrow$	$R \uparrow$	c Error $\downarrow$
B	Sequential-Q	0.23	0.10	0.46	0.46	0.79	**0.01**
	Disagreement-Q	0.36	0.10	0.76	0.39	**0.99**	**0.00**
	Random-Q	0.34	0.07	0.89	0.27	**0.98**	0.02
	LICORICE	**1.00**	**0.02**	**0.99**	**0.05**	**0.99**	**0.00**
∞	CPM [33]	1.00	0.01	1.00	0.00	0.97	0.00

Table 1: Evaluation of the reward R and concept error achieved by all methods in all environments. The reward is reported as the fraction of the reward upper bound. Unless otherwise stated, concept error is 1 - accuracy. Methods in the top section are given a budget of $B = [500, 300, 300]$, respectively; CPM is given an unlimited budget (in practice, it uses 4M, 4M, 1M concept labels respectively). For each column, the top-performing method with a *limited* budget is in bold. If there are several methods with overlapping standard deviations, we bold all of them. Full results with standard deviation are in Table 6, Appendix B.

queries for PixelCartPole-v1, DoorKey-7x7, and DynamicObstacles-5x5, respectively, as depicted in the first section of Table 1. We choose these budgets by starting from 500 and then decreasing by units of 100 until we find that LICORICE can no longer achieve 99% of the reward upper bound if we further decrease the budget. We treat the minimum budget under which LICORICE can achieve 99% of the reward upper bound as the maximum budget considered for that environment.

Comparison with Budget-Constrained Baselines To our knowledge, no previous algorithms exist that seek to minimize the number of train-time labels for concept-based RL training, so we implement our own baselines. In Sequential-Q, the agent spends all of B queries on the first B states it encounters during the initial policy rollout. In Disagreement-Q, the agent similarly spends its budget at the beginning of its learning process; however, it uses active learning with the same acquisition function as LICORICE to strategically choose the training data. In Random-Q, the agent receives B concept labels at random points in the training process using a probability to decide whether to query for a concept for each state. We show the results in the first section of Table 1. In the three environments, LICORICE performs similarly to or better than all baselines in terms of both reward and concept error. The main performance differences can be seen for PixelCartPole-v1 and DoorKey-7x7, where LICORICE achieves 100% and 99% of the optimal reward, respectively, while the second best algorithm achieves 36% and 89%, respectively. Random-Q, Disagreement-Q, and LICORICE perform similarly on DynamicObstacles-5x5, indicating that this environment is relatively simple and may not benefit from a more advanced strategy. Its simplicity may be due to the lack of concept distribution shift that is present in the other two environments. For example, in PixelCartPole-v1, the policy must be further refined to improve its estimate of the more on-policy concepts. In the initial training stages, when the policy is more random, the agent needs to estimate a wide range of possible values for the pole angle, as the pole starts upright and frequently falls. As the agent's policy improves, it requires more precise estimates of the pole angle primarily around the vertical position.

Comparison with Budget-Unconstrained Baseline To better understand the performance of our approach, we compare it against existing work that is not constrained by concept label budgets. Specifically, we implement Concept Policy Model (CPM) from previous work in multi-agent RL [33] but for the single-agent setting. This approach jointly trains the concept bottleneck and the policy, assuming unlimited access to concept labels. It represents an upper bound on concept accuracy, as the agent receives continuous concept feedback throughout learning. Surprisingly, LICORICE outperforms CPM in PixelCartPole-v1and has similar performance to CPM in the other two environments in this unfair comparison. We emphasize that CPM is given an unlimited budget, and in fact, it uses over 1M concept labels for each environment, whereas LICORICE uses 500 or fewer. We therefore answer **RQ 1** affirmatively: compared to baselines, LICORICE demonstrates both high concept accuracy (low concept error) and high reward on our three test environments, all while using substantially fewer concept labels than existing algorithms.

	PixelCartPole-v1			DoorKey-7x7			DynamicObstacles-5x5		
B	300	400	500	100	200	300	100	200	300
R	0.28	0.77	1.00	0.77	0.93	0.99	0.96	0.98	0.99
c Error	0.15	0.05	0.02	0.26	0.09	0.05	0.04	0.01	0.00

Table 2: Performance of LICORICE on all environments for varying maximum budgets. The reward is reported as the fraction of the reward upper bound. For PixelCartPole-v1, c error is MSE; for the other environments, c error is 1 - accuracy. For all environments, the reward increases and the error decreases as we increase the maximum budget. Full results with standard deviation are in Table 7, Appendix B.

4.3.2 Budget Allocation Effectiveness

To understand the extent to which LICORICE enables a reduction in human labeling effort without substantially impacting performance, we conduct experiments across three different environments with varying concept labeling budgets. We start with the baseline budget B from our previous experiments and test two additional budgets, decreasing by steps of 100. The only component of our algorithm that we vary is the number of iterations M. For each budget and environment, we report a single performance value that maximizes the sum of the relative reward and inverse concept error, both scaled to $[0, 1]$. The results of this experiment are shown in Table 2.

Our findings reveal that the impact of budget reductions (human labeling effort) varies across environments. In general, we see an increase in concept error and a decrease in reward for all environments with decreasing budget. However, both DynamicObstacles-5x5 and DoorKey-7x7 show more resilience to budget reductions, while PixelCartPole-v1 exhibits higher sensitivity. Specifically, we find that even with $B = 100$ for DynamicObstacles-5x5, LICORICE can still achieve 96% of the optimal reward. With $B = 200$, LICORICE can still achieve 93% of the optimal reward on DoorKey-7x7. In contrast, we see large decreases in reward from the budget reductions for PixelCartPole-v1. We therefore provide a more nuanced answer to **RQ 2**: LICORICE can indeed enable reductions in human labeling effort without substantially impacting performance, but the extent of this reduction is environment-dependent. In simpler environments, human effort can be dramatically reduced while maintaining near-optimal performance. However, in more complex environments like PixelCartPole-v1, the trade-off between reduced human effort and maintained performance becomes more pronounced. This finding aligns with the intuition that different environments have varying levels of complexity and would require different levels of human input to accurately capture all relevant concepts and their relationships.

4.3.3 Integration with Vision-Language Models

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. For consistency, we use the same hyperparameters for LICORICE as in Section 4.3.2. 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 described in Section 4.3.2. Here, the reduction in the number of "human" labels required translates to cost savings.

We show the results in Table 3. We find that GPT-4o can indeed serve as an annotator for some but not all environments. In DoorKey-7x7 and DynamicObstacles-5x5, LICORICE with GPT-4o labels achieves 83% and 88% of the optimal reward, respectively. The concept error rate is comparable to GPT-4o's labeling error, both evaluated on the same rollout observations, with sampling applied to the latter due to cost constraints. Additionally, the concept error gap between our trained concept model and GPT-4o almost diminishes with

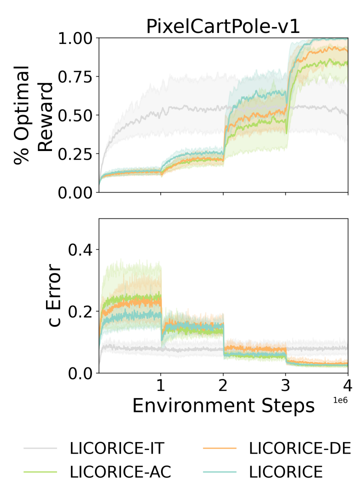

Figure 1: Training curves of reward and concept error for PixelCartPole-v1. Shaded values correspond to 95% CI, calculated using 1000 bootstrap samples. All curves in Figure 5.

B	PixelCartPole-v1 300	PixelCartPole-v1 400	PixelCartPole-v1 500	DoorKey-7x7 100	DoorKey-7x7 200	DoorKey-7x7 300	DynamicObstacles-5x5 100	DynamicObstacles-5x5 200	DynamicObstacles-5x5 300
R (LICORICE GPT-4o)	0.06	0.07	0.06	0.71	0.74	0.83	0.27	0.93	0.88
c Error (LICORICE GPT-4o)	0.17	0.25	0.25	0.45	0.33	0.31	0.20	0.13	0.13
c Error (GPT-4o)	0.19	0.22	0.24	0.31	0.31	0.30	0.16	0.12	0.13

Table 3: Performance of LICORICE with GPT-4o integrated into the loop for all environments across different budgets. We also report the concept labeling error on a random sample of 50 observations from the same rollout set used for LICORICE+GPT-4o, due to API cost constraints. Full results with standard deviation are in Table 8, Appendix B.

Algorithm	PixelCartPole-v1 $R\uparrow$	PixelCartPole-v1 c MSE $\downarrow$	DoorKey-7x7 $R\uparrow$	DoorKey-7x7 c Error $\downarrow$	DynamicObstacles-5x5 $R\uparrow$	DynamicObstacles-5x5 c Error $\downarrow$
LICORICE-IT	0.53	0.08	0.99	0.09	1.00	0.00
LICORICE-DE	0.97	0.03	0.78	0.20	0.98	0.00
LICORICE-AC	0.91	0.02	0.82	0.16	0.58	0.00
LICORICE	1.00	0.02	0.99	0.05	0.99	0.00

Table 4: Ablation study results for LICORICE in all environments. All of our components generally help achieve a better reward and lower concept error. Full results with standard deviation are in Table 9, Appendix B.

more labels. Taken together, these results suggest that LICORICE can work reasonably well with GPT-4o labels.

In PixelCartPole-v1, however, there are challenges for GPT-4o in providing accurate labels due to the continuous nature of the concepts and the lack of necessary knowledge of physical rules and quantities in this specific environment. As a result, LICORICE with GPT-4o labels achieves generally low reward across varying budgets. Interestingly, the concept error increases with more budget, likely due to the additional labeling noise introduced to the concept network. This suggests that while the VLM can handle labeling tasks to some extent, its performance in PixelCartPole-v1 is limited by the complexity of accurately labeling concepts. We also posit the mean reward drop in DynamicObstacles-5x5 at 300 budget is due to the randomness introduced by GPT-4o; however, we note that the range of standard deviation overlaps with the other two budgets. We therefore answer **RQ 3** with cautious optimism: there are indeed cases in which LICORICE could be used alongside VLMs. Generally speaking, a more human-intuitive and less complex environment is more likely to work well with VLMs by enabling clear and detailed labeling instructions in the prompt.

4.3.4 Ablation Study

Given the previous results, we now conduct ablations to confirm the effectiveness of our three main contributions: iterative training, decorrelation, and active learning. LICORICE-IT corresponds to LICORICE with only one iteration, LICORICE-DE corresponds to LICORICE without decorrelation, and LICORICE-AC corresponds to LICORICE without active learning (instead, it uses the entire unlabeled dataset for querying). We show the learning curves for PixelCartPole-v1 in Figure 1; the full set of curves is in Appendix B.

Table 4 depicts the results of our ablation study on all environments. The bottom row corresponds to the upper bound on performance by LICORICE, with all components included. All of our contributions are critical to achieving both high reward and low concept error. However, the component that most contributes to the reward or concept performance differs depending on the environment. For example, compared with LICORICE, LICORICE-IT exhibits the largest reward gap for PixelCartPole-v1; however, LICORICE-AC yields the largest reward gap for DynamicObstacles-5x5, and LICORICE-DE yields the largest gap for DoorKey-7x7. We suspect that this difference is because the concepts in DynamicObstacles-5x5 are simple enough such that one iteration is sufficient for learning, meaning that the largest gains can be made by using active learning. In contrast, PixelCartPole-v1 requires further refinement of the policy to better estimate on-distribution concept values, so the largest gains can be made by leveraging multiple iterations.

5 Related Work

Interpretable Reinforcement Learning Interpretable RL has gained significant attention in recent years [10]. One prominent approach uses rule-based methods — such as decision trees [26, 29], logic [8], and programs [30, 19] — 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 Reinforcement Learning Inspired by successes in the supervised setting [6, 25, 34], concept-based explanations have recently been incorporated into RL. One approach [7] learns a joint embedding model between state-action pairs and concept-based explanations to expedite learning via reward shaping. Unlike our work, their policy is not a strict function of the concepts, allowing our techniques to be combined to provide both concept-based explanations and a concept-based interpretable policy. Another example, CPM [33], is a multi-agent RL concept architecture that focuses on trade-offs between interpretability and accuracy. They assume that concept labels are available continuously during training. As we have shown, this approach uses over 1M concept labels in our test environments, whereas our approach reduces the need for continuous human intervention, requiring only 500 or fewer concept labels to achieve similar or better performance in single-agent environments.

Learning Concepts with Human Feedback Another line of work explores how to best leverage human concept labels but not in the RL setting, and does not focus on reducing the labeling burden. In contrast, our approach aims to reduce the concept labeling burden during *training*. One work [14] instead has users label additional information about the relevance of certain feature dimensions to the concept label. Another work [4] develops an intervention policy at prediction time to choose which concepts to request a label for with the goal of improving the final prediction. Future work could explore using these techniques alongside our method.

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 more efficiently from little to no labeled concept data. Our approach interleaves concept learning and RL training, uses an ensemble-based active learning technique to select informative data points for labeling, and uses a simple sampling strategy to better decorrelate the concept data. We demonstrated how this approach reduces manual labeling effort. Finally, 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. These include both the impacts of using VLMs in real-world applications, as well as considerations around interpretability more generally. For a more detailed discussion, please refer to Appendix C.

Limitations and Future Work Although VLMs can be successfully used for automatic labeling of some concepts, there are still hallucination issues [2] and other failure cases, such as providing inaccurate counts. We believe that future work that seeks to improve general VLM capabilities and mitigate hallucinations would also help overcome this limitation. Another exciting direction is the work in human-AI complementarity and learning-to-defer algorithms [18] to train an additional classifier for deferring concept labeling to a person when the chance of making an error is high.

Acknowledgments and Disclosure of Funding

This work was supported in part by NSF grant IIS-2046640 (CAREER). Co-author Fang is supported in part by Sloan Research Fellowship. We thank NVIDIA for providing computing resources. We thank Zhicheng Zhang for some helpful discussions, Naveen Raman for pointing us to relevant literature, and Rex Chen and Zhiyu (Zoey) Chen for their constructive comments on a draft of this work.

References

[1] Hello gpt-4o. `https://openai.com/index/hello-gpt-4o/`.

[2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] W. H. Beluch, T. Genewein, A. Nürnberger, and J. M. Köhler. The power of ensembles for active learning in image classification. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9368–9377, 2018. URL `https://api.semanticscholar.org/CorpusID:52838058`.

[4] K. Chauhan, R. Tiwari, J. Freyberg, P. Shenoy, and K. Dvijotham. Interactive concept bottleneck models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5948–5955, 2023.

[5] M. Chevalier-Boisvert, B. Dai, M. Towers, R. de Lazcano, L. Willems, S. Lahlou, S. Pal, P. S. Castro, and J. Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

[6] K. M. Collins, M. Barker, M. Espinosa Zarlenga, N. Raman, U. Bhatt, M. Jamnik, I. Sucholutsky, A. Weller, and K. Dvijotham. Human uncertainty in concept-based ai systems. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, pages 869–889, 2023.

[7] D. Das, S. Chernova, and B. Kim. State2explanation: Concept-based explanations to benefit agent learning and user understanding. *Advances in Neural Information Processing Systems*, 36, 2023.

[8] Q. Delfosse, H. Shindo, D. Dhami, and K. Kersting. Interpretable and explainable logical policies via neurally guided symbolic abstraction. *Advances in Neural Information Processing Systems*, 36, 2024.

[9] M. Espinosa Zarlenga, P. Barbiero, G. Ciravegna, G. Marra, F. Giannini, M. Diligenti, Z. Shams, F. Precioso, S. Melacci, A. Weller, et al. Concept embedding models: Beyond the accuracy-explainability trade-off. *Advances in Neural Information Processing Systems*, 35:21400–21413, 2022.

[10] C. Glanois, P. Weng, M. Zimmer, D. Li, T. Yang, J. Hao, and W. Liu. A survey on interpretable reinforcement learning. *Machine Learning*, pages 1–44, 2024.

[11] N. Grupen, N. Jaques, B. Kim, and S. Omidshafiei. Concept-based understanding of emergent multi-agent behavior. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

[12] H. Kaur, H. Nori, S. Jenkins, R. Caruana, H. Wallach, and J. 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*, pages 1–14, 2020.

[13] 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.

[14] I. Lage and F. Doshi-Velez. Learning interpretable concept-based models with human feedback. *International Conference on Machine Learning: Workshop on Human Interpretability in Machine Learning*, 2020.

[15] X.-Y. Liu, Z. Xia, J. Rui, J. Gao, H. Yang, M. Zhu, C. Wang, Z. Wang, and J. Guo. Finrl-meta: Market environments and benchmarks for data-driven financial reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1835–1849, 2022.

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

[17] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594 (7862):207–212, 2021.

[18] H. Mozannar, H. Lang, D. Wei, P. Sattigeri, S. Das, and D. Sontag. Who should predict? exact algorithms for learning to defer to humans. In *AISTATS*, 2023.

[19] S. Penkov and S. Ramamoorthy. Learning programmatically structured representations with perceptor gradients. In *Proceedings of the International Conference on Learning Representations*, 2019.

[20] E. Poeta, G. Ciravegna, E. Pastor, T. Cerquitelli, and E. Baralis. Concept-based explainable artificial intelligence: A survey. *arXiv preprint arXiv:2312.12936*, 2023.

[21] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[22] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[23] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021.

[24] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

[25] I. Sheth and S. Ebrahimi Kahou. Auxiliary losses for learning generalizable concept-based models. *Advances in Neural Information Processing Systems*, 36, 2023.

[26] A. Silva, M. Gombolay, T. Killian, I. Jimenez, and S.-H. Son. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *International conference on artificial intelligence and statistics*, pages 1855–1865. PMLR, 2020.

[27] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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

[29] N. Topin, S. Milani, F. Fang, and M. Veloso. Iterative bounding mdps: Learning interpretable policies via non-interpretable methods. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9923–9931, 2021.

[30] A. Verma, V. Murali, R. Singh, P. Kohli, and S. Chaudhuri. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pages 5045–5054. PMLR, 2018.

[31] C.-H. H. Yang, I. Danny, T. Hung, Y. Ouyang, and P.-Y. Chen. Causal inference q-network: Toward resilient reinforcement learning. In *Self-Supervision for Reinforcement Learning Workshop-ICLR 2021*, 2021.

[32] C. Yu, J. Liu, S. Nemati, and G. Yin. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021.

[33] R. Zabounidis, J. Campbell, S. Stepputtis, D. Hughes, and K. P. Sycara. Concept learning for interpretable multi-agent reinforcement learning. In *Conference on Robot Learning*, pages 1828–1837. PMLR, 2023.

[34] M. E. Zarlenga, K. M. Collins, K. Dvijotham, A. Weller, Z. Shams, and M. Jamnik. Learning to receive help: Intervention-aware concept embedding models. *Advances in Neural Information Processing Systems*, 2023.

Environment	Concept Name	Type	Value Ranges	GPT-4o Error
PixelCartPole-v1	Cart Position	Continuous	$(-2.4, 2.4)$	0.01 ± 0.00
	Cart Velocity	Continuous	$\mathbb{R}$	0.31 ± 0.12
	Pole Angle	Continuous	$(-.2095, .2095)$	0.01 ± 0.00
	Pole Angular Velocity	Continuous	$\mathbb{R}$	0.63 ± 0.14
DoorKey-7x7	Agent Position x	Discrete	5	0.36 ± 0.08
	Agent Position y	Discrete	5	0.41 ± 0.12
	Agent Direction	Discrete	4	0.28 ± 0.06
	Key Position x	Discrete	6	0.20 ± 0.03
	Key Position y	Discrete	6	0.32 ± 0.11
	Door Position x	Discrete	5	0.37 ± 0.07
	Door Position y	Discrete	5	0.32 ± 0.04
	Door Open	Discrete	2	0.38 ± 0.09
	Direction Movable Right	Discrete	2	0.30 ± 0.03
	Direction Movable Down	Discrete	2	0.19 ± 0.06
	Direction Movable Left	Discrete	2	0.33 ± 0.06
	Direction Movable Up	Discrete	2	0.20 ± 0.08
DynamicObstacles-5x5	Agent Position x	Discrete	3	0.03 ± 0.06
	Agent Position y	Discrete	3	0.04 ± 0.06
	Agent Direction	Discrete	4	0.20 ± 0.05
	Obstacle 1 Position x	Discrete	3	0.08 ± 0.02
	Obstacle 1 Position y	Discrete	3	0.19 ± 0.04
	Obstacle 2 Position x	Discrete	3	0.22 ± 0.09
	Obstacle 2 Position y	Discrete	3	0.12 ± 0.02
	Direction Movable Right	Discrete	2	0.19 ± 0.06
	Direction Movable Down	Discrete	2	0.14 ± 0.05
	Direction Movable Left	Discrete	2	0.13 ± 0.02
	Direction Movable Up	Discrete	2	0.07 ± 0.08

Table 5: Concepts and their possible values for all environments. For discrete concepts, we report the number of categories. We also provide the mean GPT-4o labeling error (MSELoss for continuous values and 1 - accuracy for discrete ones) for each single concept, averaged across 5 seeds and corresponding to the same evaluation protocols as the ones with largest budgets in Table 8. The $\pm$ [value] part shows the standard deviation. The average errors over concepts are 0.24 ± 0.06, 0.30 ± 0.02, and 0.13 ± 0.03 respectively for the three environments.

A Experimental Result Reproducibility

In this section, we provided detailed descriptions to achieve reproducibility.

A.1 Concepts Definitions

Table 5 provides more details on the concepts used in each environment, categorizing them by their names, types, and value ranges. For the PixelCartPole-v1 environment, all concepts such as Cart Position, Cart Velocity, Pole Angle, and Pole Angular Velocity are continuous. In contrast, the DoorKey-7x7 environment features discrete concepts like Agent Position (x and y), Key Position (x and y), and Door Open status, each with specific value ranges. Similarly, the DynamicObstacles-5x5 environment lists discrete concepts, including Agent and Obstacle positions, with corresponding value ranges.

We also visualize the start configurations for DoorKey-7x7 in Figure 2 to illustrate the importance of concept definitions. As shown, the concepts must be defined in such a way to allow the agent to generalize to all possible environment configurations.

A.2 LICORICE Details

In this section, we provide additional implementation details for LICORICE. The model architecture has been mostly described in the main text and we state all additional details here. The number of neurons in the concept layer is exactly the number of concepts. For continuous concept values, we

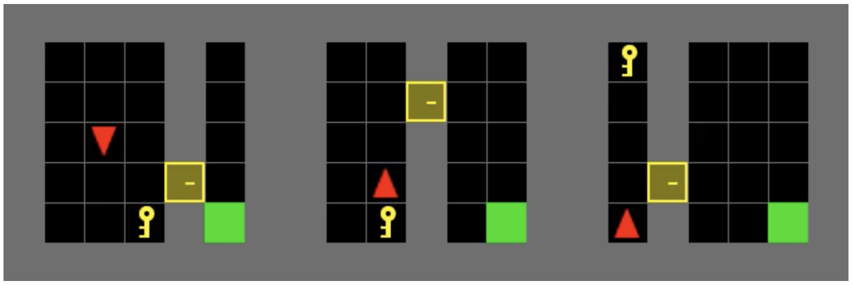

Figure 2: Example start configurations of the DoorKey-7x7 environment. This shows that concepts must general enough to apply to many different configurations. As a result, the agent cannot memorize the exact position of the door and key.

directly use a linear layer to map from features to concept values. For discrete concept values, since different concepts have different numbers of categories, we create one linear classification head for each single concept, and to predict the final action, we calculate the class with the largest predicted probability for each concept.

Value-Based Methods If we were to use a value-based method as the RL backbone, we would need to make the following changes. First, we would need to modify $V(s, a)$ or $Q(s, a)$ to include a concept bottleneck, such that $Q(s, a) = f(g(s))$. Then, we can conduct interleaved training in a similar way to LICORICE.

Feature Extractor If we use the actor-critic paradigm, we propose to share a feature extractor between the policy and value networks, shown in Figure 3. Intuitively, this choice can offer several advantages compared with using image or predicted concepts as input for both networks. Sharing a feature extractor enables both networks to benefit from a common, rich representation of the input data, reducing the number of parameters to be trained. More importantly, it balances the updates of the policy and value networks. In experiments, we observed that directly using the raw image as input for both networks complicated policy learning. Conversely, relying solely on predicted concepts for the value network may limit its accuracy in value estimation, particularly if the concepts do not capture all the nuances relevant to the value predictions.

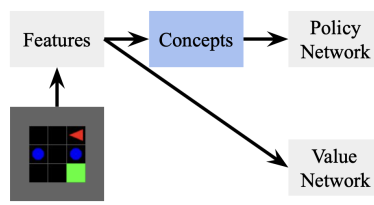

Figure 3: Architecture of our concept-bottleneck actor-critic method.

A.3 VLM Details

We detail our prompts for each environment here.

PixelCartPole-v1

> **Prompt:** Here are the past 4 rendered frames from the CartPole environment. Please use these images to estimate the following values in the latest frame (the last one):
>
> - Cart Position, within the range (-2.4, 2.4)
> - Cart Velocity
> - Pole Angle, within the range (-0.2095, 0.2095)
> - Pole Angular Velocity
>
> Additionally, please note that the last action taken was [last action].
>
> Please carefully determine the following values and give concise answers one by one. Make sure to return an estimated value for each parameter, even if the task may look challenging.
>
> Follow the reporting format:
>
> - Cart Position: estimated_value
> - Cart Velocity: estimated_value
> - Pole Angle: estimated_value
> - Pole Angular Velocity: estimated_value

DoorKey-7x7

Prompt: Here is an image of a 4x4 grid composed of black cells, with each cell either empty or containing an object. Each cell is defined by an integer-valued coordinate system starting at (1, 1) for the top-left cell. The coordinates increase rightward along the x-axis and downward along the y-axis. Within this grid, there is a red isosceles triangle representing the agent, a yellow cell representing the door (which may visually disappear if the door is open), a yellow key icon representing the key (which may disappear), and one green square representing the goal. Carefully analyze the grid and report on the following attributes, focusing only on the black cells as the gray cells are excluded from the active black area.

Detailed Instructions:

1. Agent Position: Identify and report the coordinates (x, y) of the red triangle (agent). Ensure the accuracy by double-checking the agent's exact location within the grid.
2. Agent Direction: Specify the direction the red triangle is facing, which is the orientation of the vertex (pointy corner) of the isosceles triangle. Choose from 'right', 'down', 'left', or 'up'. Clarify that this direction is independent of movement options.
3. Key Position: Provide the coordinates (x, y) where the key is located. If the key is absent, report as (0, 0). Verify visually that the key is present or not before reporting.
4. Door Position:
   - Position: Determine and report the coordinates (x, y) of the door.
   - Status: Assess whether the door is open or closed (closed means the door is visible as a whole yellow cell, while open means the door disappears visually). Report as 'true' for open and 'false' for closed. Double-check the door's appearance to confirm if it is open or closed.
5. Direction Movable: Evaluate and report whether the agent can move one cell in each specified direction, namely, the neighboring cell in that direction is active and empty (not key, closed door, or grey inactive cell):
   - Right (x + 1): Check the cell to the right.
   - Down (y + 1): Check the cell below.
   - Left (x - 1): Check the cell to the left.
   - Up (y - 1): Check the cell above.

   Each direction's feasibility should be reported as 'true' if clear and within the grid, and 'false' otherwise.

Reporting Format: Carefully report each piece of information sequentially, following the format 'name: answer'. Ensure each response is precise and reflects careful verification of the grid details as viewed.

DynamicObstacles-5x5

Prompt: Here is an image of a 3x3 grid composed of black cells, with each cell either empty or containing an object. Each cell is defined by an integer-valued coordinate system starting at (1, 1) for the top-left cell. The coordinates increase rightward along the x-axis and downward along the y-axis. Within this grid, there is a red isosceles triangle representing the agent, two blue balls representing obstacles, and one green square representing the goal. Please carefully determine the following values and give concise answers one by one:

1. Agent Position: Identify and report the coordinates (x, y) of the red triangle (agent). Ensure the accuracy by double-checking the agent's exact location within the grid.
2. Agent Direction: Specify the direction the red triangle is facing, which is the orientation of the vertex (pointy corner) of the isosceles triangle. Choose from 'right', 'down', 'left', or 'up'. Clarify that this direction is independent of movement options.
3. Obstacle Position: Identify and report the coordinates of the two obstacles in ascending order. Compare the coordinates by their x-values first. If the x-values are equal, compare by their y-values.
   (a) First Obstacle: Provide the coordinates (x, y) of the first blue ball.
   (b) Second Obstacle: Provide the coordinates (x, y) of the second blue ball.
4. Direction Movable: Evaluate and report whether the agent can move one cell in each specified direction, namely, the neighboring cell in that direction is active and empty (not obstacle or out of bounds):
   - Right (x + 1): Check the cell to the right.
   - Down (y + 1): Check the cell below.
   - Left (x - 1): Check the cell to the left.
   - Up (y - 1): Check the cell above.

   Each direction's feasibility should be reported as 'true' if clear and within the grid, and 'false' otherwise.

Reporting Format: Carefully report each piece of information sequentially, following the format 'name: answer'. Ensure each response is precise and reflects careful verification of the grid details as viewed.

A.4 Experimental Details

For the PPO hyperparameters, we set $4 \cdot 10^6$ total timesteps for PixelCartPole-v1and DoorKey-7x7, and 10^6 for DynamicObstacles-5x5. Besides that, for all environments, we use 8 vectorized environments, horizon $T = 4096$, 10 epochs for training, batch size of 512, learning rate $3 \cdot 10^{-4}$, entropy coefficient 0.01, and value function coefficient 1. For all other hyperparameters, we use the default values from Stable Baselines 3 [23].

For the concept training, we set 100 epochs with Adam optimizer with the learning rate linearly decaying from $3 \cdot 10^{-4}$ to 0 for each iteration in PixelCartPole-v1. In DoorKey-7x7 and DynamicObstacles-5x5, we use the same optimizer and initial learning rate, yet set 50 epochs instead and set early stopping with threshold linearly increasing from 10 to 20, to incentivize the concept network not to overfit in earlier iterations. The batch size is 32.

For LICORICE, we set the sample acceptance rate $p = 0.05$, the ratio for active learning $\tau = 10$, batch size to query labels in the active learning module $b = 20$, and the number of ensemble models $N = 5$. The number of iterations chosen in our algorithm is $M = 4$ for PixelCartPole-v1and $M = 2$ for the other two environments.

We use NVIDIA A6000 and NVIDIA RTX 6000 Ada Generation. Each of our programs uses less than 2GB GPU memory. For PixelCartPole-v1 and DoorKey-7x7, each run takes less than 9 hours to finish. For DynamicObstacles-5x5, each run takes less than 2 hours to finish.

For all experiments, we consistently use 5 seeds [123, 456, 789, 1011, 1213] to train the models. We then evaluate on the environment with seed 42 with 100 episodes and take the average of the reward across the episodes and the concept error across the observations within each episode.

	Algorithm	$R \uparrow$	c Error $\downarrow$
PixelCartPole-v1	Sequential-Q	0.23 ± 0.09	0.10 ± 0.04
	Disagreement-Q	0.36 ± 0.20	0.10 ± 0.04
	Random-Q	0.34 ± 0.14	0.07 ± 0.02
	LICORICE	1.00 ± 0.00	0.02 ± 0.00
	CPM	1.00 ± 0.00	0.01 ± 0.00
DoorKey-7x7	Sequential-Q	0.46 ± 0.09	0.46 ± 0.05
	Disagreement-Q	0.76 ± 0.12	0.39 ± 0.07
	Random-Q	0.89 ± 0.03	0.27 ± 0.08
	LICORICE	0.99 ± 0.01	0.05 ± 0.01
	CPM	1.00 ± 0.00	0.00 ± 0.00
DynamicObstacles-5x5	Sequential-Q	0.79 ± 0.44	0.01 ± 0.01
	Disagreement-Q	0.99 ± 0.01	0.00 ± 0.00
	Random-Q	0.98 ± 0.01	0.02 ± 0.01
	LICORICE	0.99 ± 0.00	0.00 ± 0.00
	CPM	0.97 ± 0.02	0.00 ± 0.00

Table 6: Evaluation of the reward R and concept error achieved by all methods in all environments. This is a extended table from Table 1. The reward is reported as the fraction of the reward upper bound. For PixelCartPole-v1, the c error is the MSE. For the other two environments, the c error is 1 - accuracy. The first four algorithms are given a budget of $B = [500, 300, 300]$ for each environment, from top to bottom; CPM is given an unlimited budget (in practice, it uses 4M, 4M, 1M concept labels respectively). The $\pm$ [value] part shows the standard deviation. This shows a more complete version of the results in Table 1.

	B	M	$R \uparrow$	c Error $\downarrow$
PixelCartPole-v1	300	1	0.28 ± 0.10	0.15 ± 0.07
	400	3	0.77 ± 0.22	0.05 ± 0.01
	500	4	1.00 ± 0.00	0.02 ± 0.00
DoorKey-7x7	100	1	0.77 ± 0.04	0.26 ± 0.04
	200	2	0.93 ± 0.02	0.09 ± 0.01
	300	2	0.99 ± 0.01	0.05 ± 0.01
DynamicObstacles-5x5	100	1	0.96 ± 0.02	0.04 ± 0.01
	200	1	0.98 ± 0.01	0.01 ± 0.00
	300	2	0.99 ± 0.00	0.00 ± 0.00

Table 7: Performance of LICORICE on all environments for varying budgets. We select M through grid search, optimizing R - c error without additional weighting since they are observed to have the same magnitude. The reward is reported as the fraction of the reward upper bound. For PixelCartPole-v1, c error is MSE; for the other environments, c error is 1 - accuracy. The $\pm$ [value] part shows the standard deviation. This shows a more complete version of the results in Table 2.

B Additional Results

B.1 Balancing Concept Performance and Environment Reward

In Table 6, we present an extension of the results in Table 1, including standard deviation. Our method enjoys low variance across all environments in terms of both concept error and reward.

B.2 Budget Allocation Effectiveness

In Table 7, we present an extension of the results in Table 2, including the standard deviation. As expected, as the budget increases, the standard deviation for both the reward and concept error tends to decrease. The one exception is the reward for PixelCartPole-v1. Interestingly, the standard deviation

Environment	B	M	$R\uparrow$	LICORICE+GPT-4o c Error $\downarrow$	GPT-4o c Error
PixelCartPole-v1	300	1	0.06 ± 0.02	0.17 ± 0.19	0.19 ± 0.25
	400	2	0.07 ± 0.02	0.25 ± 0.15	0.22 ± 0.15
	500	2	0.06 ± 0.01	0.25 ± 0.08	0.24 ± 0.07
DoorKey-7x7	100	1	0.71 ± 0.04	0.45 ± 0.04	0.31 ± 0.03
	200	2	0.74 ± 0.04	0.33 ± 0.01	0.31 ± 0.03
	300	2	0.83 ± 0.06	0.31 ± 0.01	0.30 ± 0.03
DynamicObstacles-5x5	100	1	0.27 ± 0.37	0.20 ± 0.03	0.16 ± 0.03
	200	1	0.93 ± 0.05	0.13 ± 0.03	0.12 ± 0.05
	300	1	0.88 ± 0.09	0.13 ± 0.03	0.13 ± 0.03

Table 8: Performance of LICORICE with GPT-4o integrated into the loop for all environments across different budgets, along with the concept labeling error of GPT-4o as a reference. We select M through grid search, optimizing R - c error without additional weighting since they are observed to have the same magnitude. GPT-4o c error is evaluated on a random sample of 50 observations from the same rollout set used for LICORICE+GPT-4o, due to API cost constraints. This shows a more complete version of the results in Table 3.

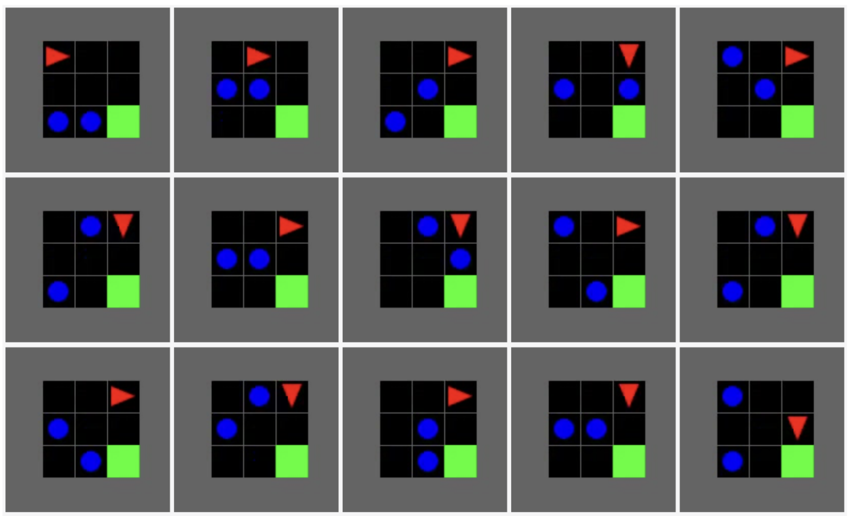

Figure 4: Concept policy model trained with concept labels provided by GPT-4o waits until the coast is clear to move to the goal. It appears to make a small mistake in the bottom left, requiring it to wait slightly longer than necessary to navigate to the goal.

is highest for $B = 400$. We suspect this is because the concept errors may be more critical here, leading to higher variance in the reward performance.

B.3 Integration with Vision-Language Models

In Table 8, we present an extension of the results in Table 3 in the main paper, including the standard deviation. Interestingly, the standard deviation for the reward obtained by using LICORICE with GPT-4o as the annotator does not always follow the same trend as shown in Table 7 (when we assume access to a more accurate human annotator). Instead, the standard deviation is relatively consistent for PixelCartPole-v1, regardless of the budget. It steadily decreases for DoorKey-7x7, as expected. However, in DynamicObstacles-5x5, we see an increase when $B = 300$. We are not sure of the cause of this. Perhaps at this point the algorithm begins overfitting to the errors in the labels from GPT-4o (the concept error rate is the same for 200 and 300 labels). Further investigation is required to understand the underlying factors contributing to this anomaly.

GPT-4o Concept Labeling Errors Table 5 list detailed concept errors for concepts in all environments. In PixelCartPole-v1, cart position and pole angle have smaller errors, while velocities require

	Algorithm	$R \uparrow$	c Error $\downarrow$
PixelCartPole-v1	LICORICE-IT	0.53 ± 0.26	0.08 ± 0.03
	LICORICE-DE	0.97 ± 0.05	0.03 ± 0.01
	LICORICE-AC	0.91 ± 0.10	0.02 ± 0.01
	LICORICE	1.00 ± 0.00	0.02 ± 0.00
DoorKey-7x7	LICORICE-IT	0.99 ± 0.01	0.09 ± 0.02
	LICORICE-DE	0.78 ± 0.08	0.20 ± 0.05
	LICORICE-AC	0.82 ± 0.10	0.16 ± 0.06
	LICORICE	0.99 ± 0.01	0.05 ± 0.01
DynamicObstacles-5x5	LICORICE-IT	1.00 ± 0.00	0.00 ± 0.00
	LICORICE-DE	0.98 ± 0.01	0.00 ± 0.00
	LICORICE-AC	0.58 ± 0.53	0.00 ± 0.00
	LICORICE	0.99 ± 0.00	0.00 ± 0.00

Table 9: Ablation study results for LICORICE in all environments, where we delete every single component to compare with LICORICE. This shows a more complete version of the results in Table 4.

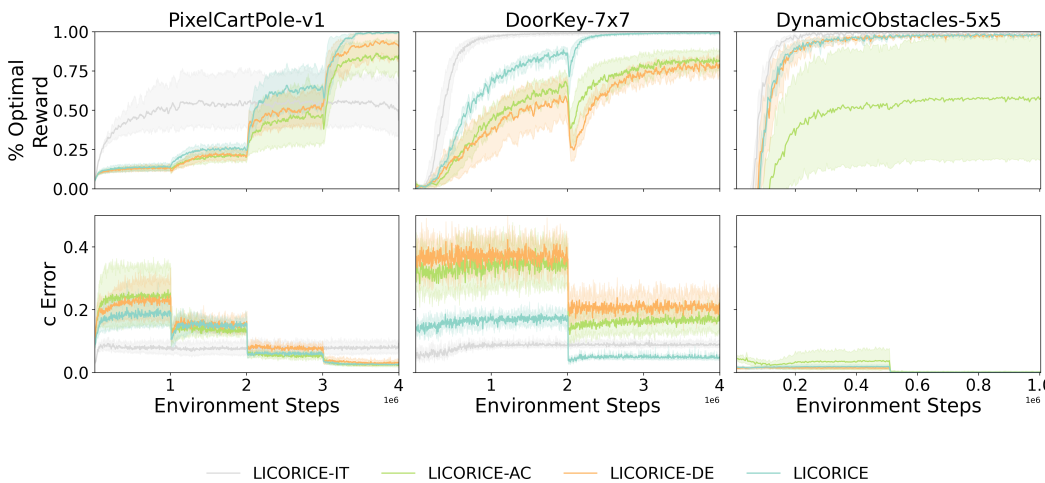

Figure 5: Learning curves for all ablations. Shaded region shows 95% CI, calculated using 1000 bootstrap samples.

understanding multiple frames and thus are harder to predict accurately. For DoorKey-7x7 and DynamicObstacles-5x5, different concepts have slightly varying concept errors, indicating visual tasks have different difficulties for GPT-4o. Agent direction has as high as around 0.3 prediction error for both DoorKey-7x7 and DynamicObstacles-5x5. Direction movable is also hard for DoorKey-7x7, with a high concept error even if it is a binary concept. We posit it requires the correct understanding of more than one particular object to ensure correctness. The concept accuracies in DoorKey-7x7 are generally higher than DynamicObstacles-5x5, suggesting GPT-4o more struggles with a larger grid.

Example Behavior of Agent with GPT-4o-labeled concepts In Figure 4, we show an example of a GPT-4o-trained RL agent on DynamicObstacles-5x5, in which the agent appears to wait until it is safe to move towards the goal: the green square. The image sequence shows the agent (red triangle) starting from its initial position and moving to the right. It then is cornered by an obstacle (blue circle), then both obstacles. In the bottom left corner frame, it appears to make a mistake by turning to the right, meaning it missed a window to escape. Finally, it moves to the goal when the path is clear in the second-to-last and last frames (bottom right). This behavior highlights that the agent may still learn reasonable behavior even if the concept labels may be incorrect (causing it to make a mistake, as in the bottom left frame).

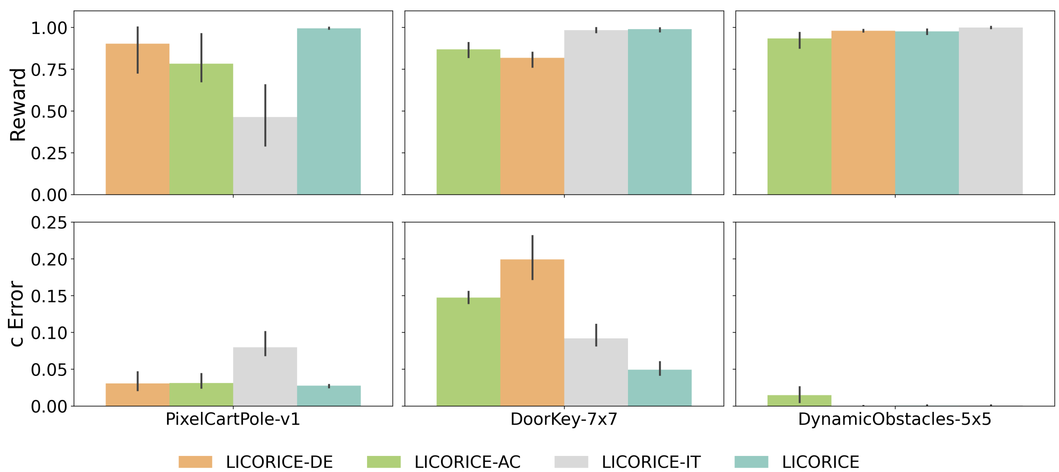

Figure 6: Performance of all ablations at the final training iteration, with black bars representing the 95% confidence interval, calculated using 1000 bootstrap samples.

B.4 Ablation

In Table 9, we present an extension of the results in Table 4, including standard deviation. Figure 5 shows all learning curves for ablations in all environments. In PixelCartPole-v1, we clearly see the benefit of iterative strategies on reward: LICORICE, LICORICE-DE and LICORICE-AC consistently increase in reward and converge at high levels, but LICORICE-IT converges at less than 50% of the optimal reward. We also see that LICORICE-IT also achieves higher concept error, indicating that it struggles to learn the larger distribution of concepts induced by a non-optimal policy. In DoorKey-7x7, all algorithms steadily increase in reward. However, we can see a dip at around 10^6 environment steps where we begin the second iteration for LICORICE, LICORICE-AC, and LICORICE-DE. Although LICORICE-IT achieves similar reward to LICORICE, LICORICE achieves lower concept error, which is beneficial for the goal of interpretability. Finally, in DynamicObstacles-5x5, LICORICE-AC lags behind the most in terms of both reward and concept error. This result indicates that the active learning component is most important for this environment.

We now inspect the performance at the last training iteration in Figure 6. Overall, we find that LICORICE performs better than or equal to the ablations on all environments. It enjoys the best reward performance on PixelCartPole-v1, while performing similarly to LICORICE-IT on DoorKey-7x7 and to LICORICE-IT and LICORICE-DE on DynamicObstacles-5x5. It exhibits the lowest concept error on PixelCartPole-v1 and DoorKey-7x7, and achieves similarly low error to LICORICE-DE and LICORICE-IT on DynamicObstacles-5x5.

In PixelCartPole-v1, LICORICE achieves the highest reward, indicating the benefits of multiple iterations, active learning, and decorrelation, which are absent in LICORICE-IT, LICORICE-AC, and LICORICE-DE, respectively. LICORICE also enjoys the lowest concept error, showcasing its ability to learn and apply concepts accurately. LICORICE-AC and LICORICE-DE perform well in minimizing concept error, while LICORICE-IT shows a relatively higher error rate, underscoring the importance of multiple iterations in some environments.

C Additional Discussion

C.1 Limitations and Future Work

While our approach has demonstrated promising results, there are several limitations to be addressed in future work. One significant challenge is the difficulty VLMs face with certain types of concepts, especially continuous variables. This limitation can impact the overall performance of concept-based models, especially in domains where continuous data is prevalent. Addressing this issue could involve developing specialized techniques or using existing tools and libraries to better complement VLM capabilities.

Another area for future improvement is the refinement of our active learning and sampling schemes. Our current method employs an disagreement-based acquisition function to select the most informative data points for labeling. While this approach is effective, there is potential for exploring more sophisticated active learning strategies, such as incorporating advanced exploration-exploitation trade-offs or leveraging recent advancements in active learning algorithms [28].

Finally, designing a concept-based representation for RL remains an open challenge. Our work provides a few illustrative examples, but the exact design of these representations can significantly impact performance, often for reasons that are not entirely clear — especially when using VLMs as annotators. Prior work [7] proposed some desiderata for concepts in RL, but future work could refine these principles, especially in the face of VLM annotators. Future work could also include systematically investigating the factors that influence the effectiveness of different concept-based representations in RL. This could involve extensive empirical studies, theoretical analyses, and the development of new design principles that guide the creation of effective concept representations. Understanding these factors better will help in creating more reliable and interpretable RL models, ultimately advancing the field and broadening the applicability of concept-based approaches in various RL tasks.

C.2 Broader Impacts

Interpretability in RL Incorporating concept learning with RL presents both positive and negative societal impacts. On the positive side, promoting interpretability and transparency in decision-making fosters trust and accountability. However, in cases where it yields incorrect results, stakeholders might be misled into trusting flawed decisions due to the perceived transparency of the model [12]. Unintended misuse could also occur if stakeholders lack the technical expertise to accurately interpret the models, leading to erroneous conclusions and potentially harmful outcomes. To mitigate these risks, an avenue for future work is developing clear guidelines for interpreting these models and tools to scaffold non-experts' understanding of the model outputs.

Using VLMs for Concept Labeling On one hand, VLMs have the potential to significantly improve the efficiency and scalability of labeling processes, which can accelerate advancements in various fields. By automating the labeling of large datasets, VLMs can help reduce the time and cost associated with manual labeling. However, there are important ethical and social considerations to address. One major concern is the potential for bias in the concept labels generated by VLMs. If these models are trained on biased or unrepresentative data, they may perpetuate or even amplify existing biases, leading to unfair or discriminatory outcomes. This is particularly problematic in sensitive applications like hiring, lending, or law enforcement, where biased decisions can have significant negative impacts on individuals and communities. Furthermore, there are privacy concerns related to the data used to train VLMs. Large-scale data collection often involves personal information, and improper handling of this data can lead to privacy violations. To mitigate these risks, future work could include developing robust data governance frameworks to protect individuals' privacy and comply with relevant regulations.