Strategic Object Oriented Reinforcement Learning

Ramtin Keramati*
Stanford University
Stanford, CA 94305
keramati@cs.stanford.edu

Jay Whang*
Stanford University
Stanford, CA 94305
jaywhang@cs.stanford.edu

Patrick Cho*
Stanford University
Stanford, CA 94305
patcho@cs.stanford.edu

Emma Brunskill
Stanford University
Stanford, CA 94305
ebrun@cs.stanford.edu

Abstract

Humans learn to play video games significantly faster than state-of-the-art reinforcement learning (RL) algorithms. Inspired by this, we introduce strategic object oriented reinforcement learning (SOORL) to learn simple dynamics model through automatic model selection and perform efficient planning with strategic exploration. We compare different exploration strategies in a model-based setting in which exact planning is impossible. Additionally, we test our approach on perhaps the hardest Atari game *Pitfall!* and achieve significantly improved exploration and performance over prior methods.

1 Introduction

The coupling of deep neural networks and reinforcement learning has led to extremely exciting advances, enabling reinforcement learning agents that can reach human-level performance in many Atari2600 games [19]. However such agents typically require hundreds of millions of time steps to learn to play well. As recently noted [16], this is in sharp contrast to people, who can learn to many things, including Atari games, extremely quickly. Prior investigations into how humans learn to play Atari highlights peoples' ability to generalize from few examples, use higher level representations (specifically objects) and suggest people may strategically explore the dynamics models of those objects, that they can then use to compute plans likely to yield high reward [28].

An important question is whether we can define algorithms that can similarly leverage such strategies to enable vastly more efficient reinforcement learning. In particular, we hypothesize that the intersection of three features may be sufficient to start to enable such success: leveraging abstract object-level representations, learning (often inaccurate) models of the world dynamics that can be learned quickly and support fast planning, and strategic model-based exploration using lookahead planning.

Strategic exploration methods have been studied in great detail in the tabular setting. However, extending these methods to large MDPs remain difficult. Recent extensions of tabular optimistic bonuses (e.g. MBIE [25]) to deep model-free RL methods [2, 22, 26] show substantially improved performance in many environments over ϵ-greedy exploration. Other model-free strategic exploration [20, 1, 8] deep RL methods have also shown promising results. These methods still require millions of frames and struggle in domains that involve sparse delayed reward.

A challenge with many of these methods is propagating the optimistic reward bonuses to encourage exploration, a challenge that should be addressed by performing lookahead planning using model-

*These authors contributed equally to this work

Preprint. Work in progress.

based RL, such as by using UCT algorithm [15]. Unfortunately, so far model based deep RL has not matched the impressive performance observed by its model free DRL counterparts. This is likely in part because learning accurate models in complicated domains can require large models that take a lot of data to train, and if the models are poor, such errors are well known to compound during planning.

Indeed, some prior work tries to use object level representations to provide a key inductive bias for accelerating learning good representations, dynamics and reward for reinforcement learning [9, 23, 5]. The vast majority of this work does not couple this effort with strategic exploration methods, which is perhaps why the majority of this work has not observed substantial improvements over DRL methods. Our work is inspired by an important exception to this, the DOORMAX algorithm[6] which performs strategic R-max [4] like exploration to learn a logic-like representation of the dynamics in the *Pitfall!* game, and successfully learns to quickly pass the first room. However such work assumes planning can be done exactly, which is not tractable for full Atari games. Such work also assume that we can rely on a simple prior on the dynamics model classes that is insufficient for general Atari games, including the other rooms in *Pitfall!*.

In this paper we introduce Strategic Object Oriented Reinforcement Learning (SOORL) and make three key contributions. First we investigate the impact of exploration strategies using model-based RL when it is impossible (due to real-time or computational constraints) to perform exact planning. Specifically we investigate this in the context of using simple MCTS: while the planning approach could be further improved, MCTS is an extremely popular approach and our results highlight that optimistic strategies can be substantially better than Thompson sampling when doing RL using MCTS as the planner. Second we introduce a new object oriented, model-based, optimistic RL algorithm (SOORL). Our algorithm takes in simple action macros (of the form "act and then wait" identical to those defined in prior work [6]) that may mimic human performance due to reaction time, and leverages an inductive prior that there should exist simple deterministic models of the world dynamics. The algorithm performs automatic model selection and performs optimism under uncertainty planning for the selected models. Third, while we designed this algorithm to be applicable to all Atari games, we chose to evaluate it on *Pitfall!* since that is perhaps the hardest Atari2600 game, exhibiting extremely sparse reward. To our knowledge our approach is the first method to achieve positive reward on this game without human demonstrations.

2 Related Work

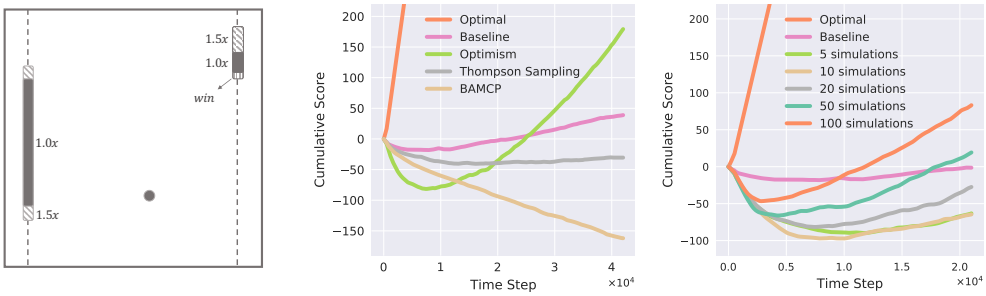
Recent advances in Deep Reinforcement Learning [19, 29, 18] have successfully extended tabular setting RL algorithms like Q-learning [30] to large state space MDPs. In particular, these algorithms have achieved either human or super human performances in many Atari games [3] . However, when compared with humans [28, 16], these methods require orders of magnitude more samples [28].

Strategic exploration has been extensively studied in the tabular setting. Most of these techniques use the notion of Optimism in Face of Uncertainty (OFU) to achieve strong theoretical guarantees [4, 25, 14, 21]. However, these methods do not scale well to large MDPs and often result in poor sample complexity. To tackle this problem, [2] proposed an extension of count based exploration to large MDPs. They do so by assigning an exploration bonus that is inversely proportional to a pseudo-count. [22] used neural density models and [26] achieved generalization over pseudo-counts by using simple hash functions. Despite good asymptotic performance in hard exploration sparse reward games like Montezuma's Revenge, these methods still require an enormous amount of data to learn.

Bayesian RL methods also provide an effective balance of exploration and exploitation [10]. However, these methods remain computationally intractable in large state spaces. [20, 1, 8] proposed an extension of these methods to large MDPs. However, these methods are unable to show substantial improvement in hard exploration, sparse reward environments.

Model based RL can improve sample efficiency. Bayes Adaptive MDP [7] is a powerful tool for combining posterior sampling exploration and model based planning. However, these methods are generally intractable in large MDPs. BAMCP [12] proposed a tractable sample-based method for approximate Bayes-optimal planning with root and lazy sampling over parameters of the state model.

Perhaps, the most related line of research is object oriented representation for RL. Model free methods to use object representation [9, 23, 5] fail to scale to large MDPs and do not leverage object



(a) Pong Prime environment (b) Different exploration methods (c) Optimism based exploration

Figure 1: Pong Prime

representation for strategic exploration. OOMDP [6] defines a notion borrowed from relational MDPs [11], and uses objects to learn models and perform model based planning. The main difference between our approach and other object oriented approaches is that we perform scalable planning with strategic exploration by leveraging objects to learn simple dynamics models.

3 Object Representation

Consider a finite horizon Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ the transition function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, and γ the discount factor. The goal of the RL agent is to maximize the expected discounted reward $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$ following a policy π. Additionally, inspired by human visual perception, we assume the existence of an object extractor function $f : \mathcal{S} \to \mathcal{O}$ that extracts the objects in state s.

Similar to OOMDP [6], we define a set of object classes $\mathcal{C} = \{c_1, \ldots, c_n\}$ where each class has a set of attributes $\{c.a_1, \ldots, c.a_m\}$. Each state s consists of objects $f(s) = \{o_1, \ldots, o_k\}$ where each object $o_i \in \mathcal{C}$. The state of an object is defined by the value assignment to its attributes. Finally, the state s of the underlying MDP is the union of all object states $\cup_{i=1}^{k} o_i$.

We define the interaction function $I : \mathcal{O} \times \mathcal{O} \to \{0, 1\}$ to be an indicator that determines if two objects are interacting with each other. For simplicity, we make three assumptions: first, that this interaction function is known; second, objects from the same class share the same transition function; and third, each object's next state is dependent on at most pairwise object interactions and action. An object's successor state is determined by a standalone transition function $T_c(o, a)$ or a pairwise transition function $T_{c_i,c_j}(o_i, o_j, a)$ if $I(o_i, o_j) = 1$.

4 Exploration with Imperfect Planning

Planning in a MDP with known dynamics can be efficiently done by focusing on promising branches of state-action tree using the UCT algorithm [15]. However, learning a model sufficient for planning can be hard, if not impossible, for a MDP with large state space. Object representation allows us to learn a simple predictive model of the dynamics for each object class and also allows us to perform strategic exploration (e.g. posterior sampling and optimism in the face of uncertainty).

Algorithm 1 describes object-level planning with strategic exploration. At each state s, we select the appropriate distribution over models for the corresponding object representation $f(s)$ and use UCT to pick the best action (section 5). This approach naturally lends itself to three different methods of exploration: **Thompson Sampling** [27], by sampling a model at the beginning of planning (equivalent to setting $K = 1$ in Algorithm 1); **BAMCP** [12], by sampling a model for each simulation (equivalent to setting $L = 1$ in Algorithm 1); and **optimism based exploration** by planning K times each with a new sampled model and acting greedily according to the maximum Q value for each action across different models. We compared these methods to **Baseline**, which uses a MLE model with UCT algorithm and no exploration.

Algorithm 1 Planning

procedure PLANNING(s, K, L)
input: s state, K number of models, L number of simulations
  $o \leftarrow$ detectObjects(s)
  $\mathcal{T}, \mathcal{R} \leftarrow$ selectModels(o)
  for k in 1:K **do**
    $T \sim \mathcal{T}$, $R \sim \mathcal{R}$
    for l in 1:L **do**
      search($T, R, o, 0$)
    end for
  end for
  return $\underset{a}{argmax}\ Q(s, a)$
end procedure

procedure SEARCH(T, R, o, d)
input: T transition function, R reward function, O object state, d current depth
  if $d \geq MaxDepth$ **then return** 0
  $a \leftarrow \underset{a}{argmax}\ Q(o, a) + c\sqrt{\frac{logN(o)}{N(o,a)}}$
  $o' \leftarrow T(o, a)$
  $r \leftarrow R(o, a)$
  $\tilde{R} \leftarrow r + \gamma$ search($T, R, o', d + 1$)
  $N(o, a) \leftarrow N(o, a) + 1$
  $N(o) \leftarrow N(o) + 1$
  $Q(o, a) \leftarrow Q(o, a) + \frac{\tilde{R} - Q(o,a)}{N(o,a)}$
  return $\tilde{R}$
end procedure

We now compare these approaches in a challenging exploration setting. We do this in order to better understand how strategic exploration methods work when using approximate model-based planning, as a key building block to tackling challenging sparse reward Atari domains. To do so, we introduce variant of the game Pong, Pong Prime (Fig. 1(a)). Dynamics of this game is similar to Pong with minor tweaks that make the game significantly harder. The enemy paddle is made 3 times larger than the player paddle, so it is impossible to score a point by simply hitting the ball back at the normal speed. Additionally, the top and bottom 10% percent of the enemy paddle hit the ball back at 1.5 times the normal speed so that the enemy paddle is even more powerful. Similarly, the player paddle also consists of 3 regions with distinct behavior. The *top region* takes up the top 50% of the paddle and hits the ball back at 1.5 times speed. The *middle region* takes up the middle 45% of the paddle and hits the ball back at normal speed. Finally, the *lower region* covers remaining 5% and instantly wins a point for the player. This configuration is set up so that it is difficult but not impossible for the player to score using the top region (scoring on average around 5% of the time the ball bounces off the top region). In this setting, the optimal policy is to always hit the ball with the lower region of the ball. The game is deterministic and model free methods with ϵ-greedy exploration (e.g. DDQN) consistently loses the game with lowest possible score across 5000 episodes.

The correct model class for dynamics of each paddle is a linear model with 3 action history. We assume that the learned dynamics model uses this true model class (i.e. performs linear regression on the transitions we observe). Figure 1(b) compares the performance of different exploration strategies to the baseline, which performs UCT with the MLE model. We perform 500 total tree searches for all runs in Figure 1(b) (i.e. Thompson Sampling uses 500 simulations and 1 model, BAMCP 1 simulation and 500 models, optimism 100 simulations and 5 models).

Both BAMCP and Thompson Sampling perform worse than using the MLE model. We hypothesize this is critically due to performing approximate planning, as we know that in the limit of infinite simulations that BAMCP is guaranteed to converge to the optimal Bayes adaptive solution [12]. Similarly, with the true problem depth and enough simulations, we should compute the exact value for the model sampled with Thompson sampling, and there are also strong guarantees that such a method will converge to the optimal policy. However in practice, in large domains or domains with real-time constraints, the amount of computation and therefore the quality of the computed plan, will be significantly limited. In particular, if it is infeasible to use a depth that mimics the game horizon, or perhaps even to reach a local reward, then Thompson sampling approaches may suffer. This is because TS methods sample a model, which means that parts of the model may be overly optimistic and others may be pessimistic. If we are doing a limited number of simulations using MCTS, then we may not go down branches of the tree that "observe" the optimistic parts of the sampled model. This means that the resulting computed estimates of the Q value at the root node may not be optimistic, which in practice is often a key part of proofs of the effectiveness of TS methods, and very helpful empirically. BAMCP will have similar challenges but suffers further in this domain because the

true domain is deterministic. This means that for TS, optimism, and the MLE approaches, the tree constructed will only have one child node for any sampled action (the deterministic next state). In contrast, BAMCP samples a different deterministic model at each simulation, and for the same action node, those models may each predict different, deterministic, next states. This means BAMCP must potentially build a tree of size O($(|A|M)^H$) when sampling M different models, in contrast to the other methods that build a tree of at most size O($|A|^H$).

Optimism-based exploration significantly outperforms other approaches. We suspect it is more robust to approximate planning, since optimism is built into *every* node, allowing it to distinguish even locally between actions that may need exploration, in absence of observing long delayed reward. To find the optimal policy still requires significant lookahead and careful planning.

Indeed as we demonstrate in Figure 1(c) for the optimistic method, as planning power increases through more simulations, the performance of optimism-based exploration also increases. We expect that with sufficient computations the optimstic method should eventually learn the optimal policy for this domain.

5 Strategic Model Based Reinforcement Learning

Besides the challenge of doing strategic exploration, another main challenge of performing efficient model based planning is learning accurate state and reward models. Learning accurate state and reward models is critical for long horizon planning as even a small bias in the model can introduce catastrophic compounding errors that make planning impossible.

5.1 Learning

Although state and reward models can be modelled by functions as general as neural networks, using such complicated functions can make performing strategic exploration difficult. Moreover, such functions require much more data to train. Given that we are planning at an object level, we hypothesize that even simple models, such as linear and discrete count based models, give sufficient accuracy for planning. More importantly, to ensure "sufficient accuracy in planning", we further require that these models predict transitions and rewards in a deterministic fashion.

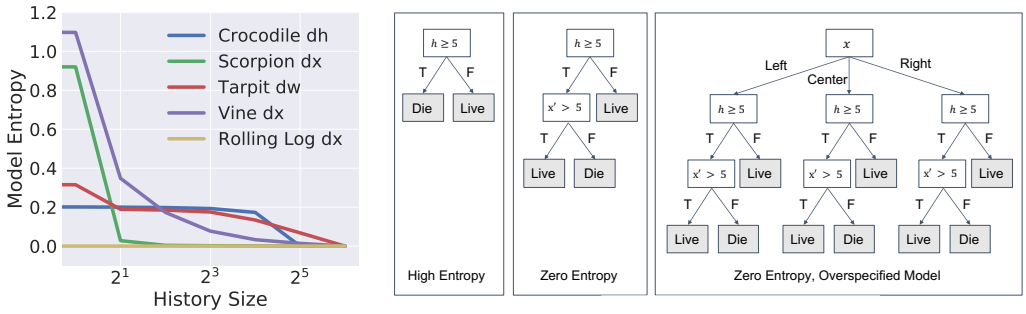
To ensure deterministic transitions, we consider the class of functions $\mathcal{F}_t = \{f_t^1, \ldots, f_t^n\}$, where each f_t^i is a count-based model of the dynamics for an object. Each function stores the count of every output based on a different set of input features with given history t. The simplest model in $\mathcal{F}_1$ is f_1^1, which uses one history with null input. For example, for a falling object with steady state velocity, such a model is sufficient as we can predict displacement δx and δy without any input. On the other hand, f_1^n, which uses one history and the most complex set of features, is the most complicated model in the class $\mathcal{F}_1$. In terms of objects, the most complex set of input features that we consider is the union of the object's state features, relative state features with respect to an interacting object, and action.

The goal then is to choose the simplest model that achieves deterministic transitions within $\mathcal{F}_t$. To do so, we compute the entropy of the observed data for each function $H(f_t^k) = -\sum_{x_i} p(x_i) \log(p(x_i|f_t^k))$ where the summation is over all the observed data. We choose the simplest model that has entropy less than a predefined threshold ϵ_{ent}. If none of the models in $\mathcal{F}_t$ satisfy the entropy threshold, we increase t through an exponential back off scheme. Concretely, we increase the history to the next exponent of 2. Figure 2(a) shows an example of the impact of this exponential back off scheme. With sufficient history, the entropy of the model eventually drops to zero. We use the same approach and same class of models for reward functions.

5.2 Planning

Each state and reward model described in section 5.1 is a multinomial distribution with Dirichlet prior. At each planning step in Algorithm 1, we compute a posterior distribution over state and reward given the current input features, and sample K different models based on some exploration method. Based on our results in section 4, we use optimism based exploration method.

Additionally, count based models allow us to efficiently perform the knows what it knows (KWIK) [17] scheme for exploration. Concretely, if our algorithm queries the state or reward model with a



(a) Model entropy

(b) Example trained models

Figure 2: Model selection

previously unseen input, we consider the resulting state as a state with R_{max} reward. R_{max} reward is also considered for any previously unseen interactions. Having R_{max} reward for unseen object interactions encourages the agent to explore the interaction and learn the reward and state model for interaction of the objects.

6 Pitfall!

Pitfall! is an Atari2600 environment where the goal is to have the agent traverse through multiple rooms (255 in total) while collecting rewards and avoiding obstacles. It is arguably the hardest Atari2600 game [13] due to its large map and sparse positive rewards that necessitate efficient exploration and long-horizon planning. Even the closest positive reward from the initial location is 7 rooms away and requires passing through several types of obstacles, each with a unique behavior. The ϵ-greedy exploration strategy completely fails in this environment, and more recent count-based exploration [2] does not show much performance boost due to the sparsity of positive reward. Pitfall is difficult even for human players without prior knowledge of the game – [13] reports that human performance varies from 3662 to 47821 points, whereas for other hard Atari games, this variation is much smaller (e.g. from 32300 to 34900 for Montezuma's revenge).

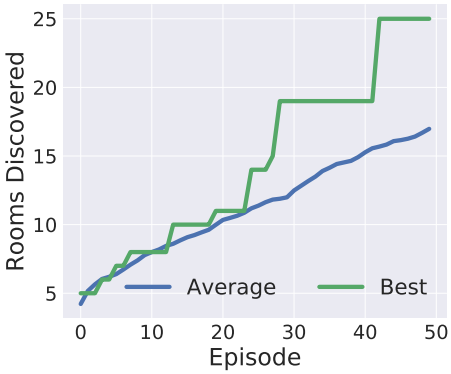
6.1 Learning State and Reward Models

Firstly, following [6], we use the notion of meta actions to simplify the model learning process. A meta action $\hat{a} \in \hat{\mathcal{A}}$ is defined as a fixed sequence of low level actions $\hat{a} : \{a_1, \ldots, a_l\}, a \in \mathcal{A}$. For Pitfall, we used the simplest form of meta-actions $\hat{a}_i = \{a_i, null \times k\}$ that is a low level action followed by k no action.

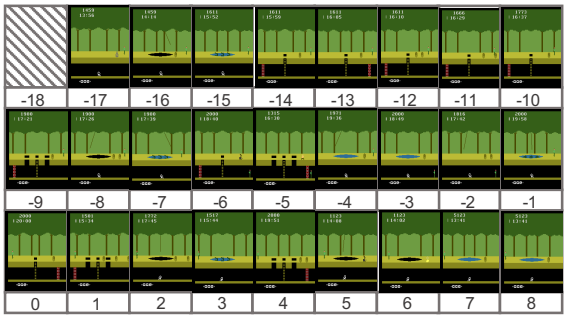
It is important to note that while the notion of meta actions allows for simpler state models, the state model still needs to learn all the low level transitions, since these transitions are necessary for planning. For example, when the agent jumps and tries to catch the vine, the algorithm needs to predict the interaction with the vine in the middle of the meta action in order to use the correct object interaction model.

Next, the object attributes that we consider in Pitfall are attributes that can be extracted from standard object detection algorithms in Computer Vision. Specifically, we extract bounding boxes for each object. Using these object detections and meta-actions, we learn the state and reward models online for each object based on the method described in Section 5. The features used are a cartesian product of object size (w, h), object location (x, y) and object intersection (x', y'). Ignoring null input, this cartesian product results in 7 different feature sets.

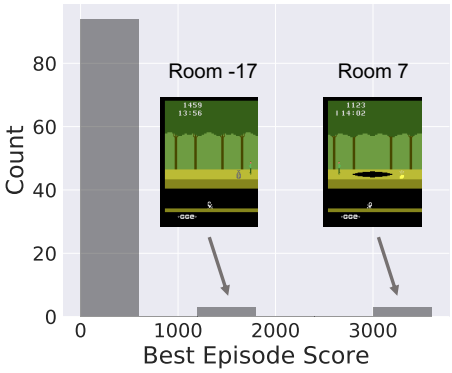
Figure 2(a) shows the entropy of the best model in the model class for different history lengths. There is a clear drop in the entropy of the model as the history length increases and the model becomes deterministic. Furthermore, for a given history, we pick the simplest model that achieves a sufficiently low entropy. For example, for the crocodile obstacle, the agent lives as long as either the crocodile's mouth is closed $(h \leq 5)$ or the agent is standing on the head of the crocodile $(x' > 5)$. Figure 2(b) shows that given a history of one, a model that only uses object size as a feature will result in high



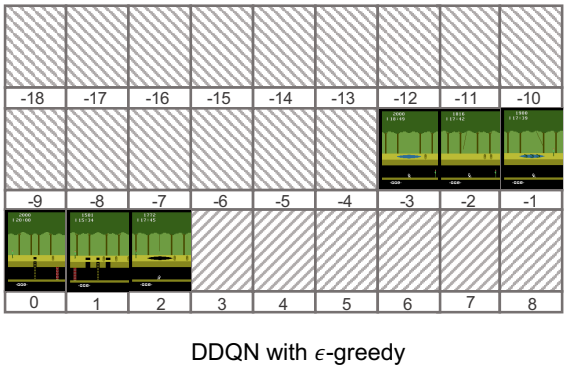
(a) Number of Discovered Rooms Per Episode



(b) Rooms Discovered via Strategic Exploration



(c) Histogram of Best Episode Scores



(d) Rooms Discovered via DDQN

Figure 3: Performance of SOORL on *Pitfall!*

entropy while a model that uses all the features is overspecified, and results in the model having to learn the three crocodiles separately. Using the approach described in Section 5, we select the model in the middle, which achieves zero entropy while generalizing over all the crocodiles.

The same approach is used to learn the reward model. In Pitfall, the reward model class is simple and requires zero input for all models to achieve sufficiently low entropy.

6.2 Exploration

Following the results in section 4, we apply optimism as our exploration scheme of choice. Leveraging KWIK, at each step of planning, if we encounter an unseen state or an unseen interaction, we consider the reward of that state to be R_{max}. Doing so encourages the agent to leverage its uncertainty over its state and reward models to learn the models quickly.

Additionally, we use count-based optimism. This is done by splitting the screen into $N \times M$ grids and keeping a count of the number of times the agent visits each grid. The agent is given a reward bonus that is proportional to a decaying function of the count on the grid it visits. To encourage exploratory behavior during the learning of state and reward models, we used the first twenty episodes as "warm-up" and reset the counts at the beginning of each episode.

6.3 Performance and Discussion

Figure 3(a) shows an increasing number of rooms being discovered across episodes. On average, the agent discovers 17 rooms within 50 episodes. The best out of 100 runs discovers 25 rooms within 50 episodes. Figure 3(b) shows all the 26 rooms that were discovered across all 100 runs. On the other hand, DDQN with ϵ-greedy exploration visits at most 6 rooms.

Method	SOORL‡	**SOORL**$^{\dagger *}$	DQfD†	Count Based†	A3C†	DQN†
Performance	-281.07 $\pm$ 395.56	**80.52**	50.8	-259.09	-6.98	-86.85

Table 1: Comparison of our method (SOORL) to state-of-the-art algorithm, † is evaluation time performance, ‡ is training time performance and $^{\dagger *}$ is the average of the best episode for each run (for our algorithm). We expect $^{\dagger *}$ and † to be the best performance of each algorithm, averaged across runs.

Table 1 compares our method to other state-of-the-art algorithms. Results of count-based [2], DQfD [13], A3C [18] and DQN [19] are reported at the time of evaluation and we expect these results to be close to their best performance. Our average score across all episodes and all runs is -281.07, which is roughly on par with count-based evaluation. Our average score for the best episode across all runs in 80.52, which is higher than all scores that were reported at the time of evaluation. Moreover, our method manages to get 6 positive rewards, 3 of which are situated in room 6 and another 3 which are situated in room -17. To the best of our knowledge, this is the first approach which manages to get positive rewards on *Pitfall!* without human demonstrations. Moreover, from video demonstrations shown by DQfD, the agent seems to only get the reward in room 6 and not the reward in room -17. In comparison, our approach explores both the left and right side of the map, and gets the rewards on both sides of the map equally often. Sample videos of the agent reaching the two closest positive rewards can be found here: `https://youtu.be/GvenPZMJiTg` (4000 reward) `https://youtu.be/74F-ta5LyuA` (2000 reward)

An immediate obvious improvement would be to incorporate an estimate of the leaf node value during tree search. This addition was critical to the success and computational efficiency of earlier MCTS methods, such as on the game Go [24]. To see why this is crucial if computation is bounded, consider that even if the agent discovers rewards in either room 6 or room -17, the agent later may die and be reset back to the initial starting room. In our current implementation, at this point the agent's forward search planning is depth-limited and is not currently sufficient for the agent to predict that it can reach those rewards again. Incorporating an estimate of the future value at the leaves will allow the agent to circumvent this problem while avoiding prohibitive lookahead planning.

7 Conclusion and Future Work

There are many exciting directions for future work. Here we highlight just a few:

Robust planning: One important challenge in model-based RL is making planning robust to model inaccuracy. Identifying the right model class is a nontrivial task, and a wrong model class can easily introduce a catastrophic error in long-horizon prediction that prohibits the use of tree search algorithms like UCT.

Value approximation: In this approach, we did not use a value function at the leaf nodes during UCT. Having such function could allow our *Pitfall!* agent to explore the state space more systematically by incorporating the value estimates of states it has previously visited. We believe that learning a value function or an action-value function at the leaf node can significantly boost planning efficiency.

To conclude, we presented an object-oriented framework that allows the RL agent to quickly learn to explore in an environment with large state space and sparse reward. Our work combines object oriented representation, automatic model selection that biases towards simple deterministic models, and strategic exploration using MCTS and optimism. We demonstrate that optimistic planning may be particularly beneficial when planning is necessarily approximate. We also demonstrate the first, to our knowledge, approach that can obtain positive reward on Pitfall without human demonstrations.

References

[1] Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through bayesian deep q-networks. *arXiv preprint arXiv:1802.04412*, 2018.

[2] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[3] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[4] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[5] Luis C Cobo, Charles L Isbell, and Andrea L Thomaz. Object focused q-learning for autonomous agents. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1061–1068. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[6] Carlos Diuk, Andre Cohen, and Michael L Littman. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 240–247. ACM, 2008.

[7] Michael O'Gordon Duff and Andrew Barto. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massachusetts at Amherst, 2002.

[8] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

[9] Marta Garnelo, Kai Arulkumaran, and Murray Shanahan. Towards deep symbolic reinforcement learning. *arXiv preprint arXiv:1609.05518*, 2016.

[10] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

[11] Carlos Guestrin, Daphne Koller, Chris Gearhart, and Neal Kanodia. Generalizing plans to new environments in relational mdps. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1003–1010. Morgan Kaufmann Publishers Inc., 2003.

[12] Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, pages 1025–1033, 2012.

[13] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, et al. Deep q-learning from demonstrations. *arXiv preprint arXiv:1704.03732*, 2017.

[14] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

[15] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[16] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

[17] Lihong Li, Michael L Littman, and Thomas J Walsh. Knows what it knows: a framework for self-aware learning. In *Proceedings of the 25th international conference on Machine learning*, pages 568–575. ACM, 2008.

[18] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[20] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

[21] Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning. *arXiv preprint arXiv:1607.00215*, 2016.

[22] Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Rémi Munos. Count-based exploration with neural density models. *arXiv preprint arXiv:1703.01310*, 2017.

[23] Melrose Roderick, Christopher Grimm, and Stefanie Tellex. Deep abstract q-networks. *arXiv preprint arXiv:1710.00459*, 2017.

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

[25] Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[26] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2750–2759, 2017.

[27] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[28] Pedro A Tsividis, Thomas Pouncy, Jacqueline L Xu, Joshua B Tenenbaum, and Samuel J Gershman. Human learning in atari. 2017.

[29] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 16, pages 2094–2100, 2016.

[30] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.