TRL: DISCRIMINATIVE HINTS FOR SCALABLE REVERSE CURRICULUM LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep reinforcement learning algorithms have proven successful in a vast variety of domains. However, tasks with sparse rewards remain challenging when the state space is large. Goal-oriented tasks are among the most typical problems in this domain, where a reward can only be received when the final goal is accomplished. In this work, we propose a potential solution to such problems with the introduction of an experience-based tendency reward mechanism, which provides the agent with additional hints based on a discriminative learning on past experiences during an automated reverse curriculum. This mechanism not only provides dense additional learning signals on what states lead to success, but also allows the agent to retain only this tendency reward instead of the whole histories of experience during multi-phase curriculum learning. We extensively study the advantages of our method on the standard sparse reward domains like Maze and Super Mario Bros and show that our method performs more efficiently and robustly than prior approaches in tasks with long time horizons and large state space. In addition, we demonstrate that using an optional checkpoint scheme with very small quantity of key states, our approach can solve difficult robot manipulation challenges directly from perception and sparse rewards.

1 INTRODUCTION

Reinforcement learning (RL) aims to learn the optimal policy of a certain task by maximizing the cumulative reward acquired from the environment. Recently, deep reinforcement learning has enjoyed great successes in a variety of domains with short-term and dense reward feedback (Mnih et al., 2016) (e.g. Atari games, TORCS). However, many real world problems are inherently based on sparse, binary rewards, where the agent needs to travel through a large number of states before a success (or failure) signal can be received. For instance, in the grasping task of a robotic arm, the most accurate task reward would be a binary reward only if the agent successfully picked up the the object or not. This kind of sparse-reward, goal-oriented tasks are considered the most difficult challenges in reinforcement learning. That is, environment only provides a final success signal when the agent has accomplished the whole task. The search space of these tasks may exponentially expand as state chain extends, which adds to the difficulty of reaching the final goal with conventional reinforcement learning algorithms.

There have been multiple lines of approaches for tacking such sparse reward problems, including the ideas based on intrinsic motivation (Stadie et al., 2015; Bellemare et al., 2016; Houthooft et al., 2016), hierarchical reinforcement learning (Florensa et al., 2017a; D.Kulkarni & Karthik Narasimhan, 2016) and curriculum learning (Karpathy & Van De Panne, 2012; Sukhbaatar et al., 2017). Recently, a particularly promising approach is the idea of reverse curriculum generation learning (Florensa et al., 2017b). By reversing the traditional reinforcement learning process and gradually expanding the set of start states from the goal to the starting point, this method does achieve substantial progress in some simulation tasks. However, the newly sampled start set mixes old and new states, which leads to inefficiency in training, as the agent spends a large amount of time reviewing rather than learning new skills. This problem aggravates in large state space tasks where

the expansion could be really slow, and the storing of history data is also impracticable due to the memory limitation.

To overcome the drawbacks mentioned above, we propose a Tendency Reinforcement Learning (TRL) architecture based on a key idea: the agent should be targeted at learning what it cannot yet do, while its former experience can provide tendency hints to stabilize and accelerate the training process. To achieve this, a dense binary reward (the "tendency") is designed to provide a rough judgment on the success tendency of each state, and our agent learns from hybrid rewards that combine the final goal reward with the hints outputted by tendency reward. We define a set of start states as a phase, and continue extending it from the final goal, until it reaches where the task starts. In each phase, a tendency reward provides hints for the agent and influences its training. Between phases, the tendency reward is updated using experience of the agent. This mutual promotion between the agent and tendency reward contributes to a dramatic improvement in training efficiency.

In this paper, we introduce three novel techniques: First, we propose a hybrid reward design that combines the final reward with tendency hints. Secondly, with the introduction of a phase administrator, the agent is able to focus on exploiting new states, which increases the training efficiency. Lastly, we develop an optional checkpoint scheme, that is, our framework is also compatible with additional checkpoints with no strict accuracy requirement, as long as they are effective at reducing irrelevant search space.

The major contribution of this work is that we present a reliable tendency reinforcement learning method that is capable of training agents to solve large state space tasks with only final reward. In our experiments, we apply and test the model in diverse domains, including a maze game, Super Mario Bros, a robotic grasping task, a crane conveyance task and a pick-and-place challenge, where it proves more efficient and competitive than the prior works.

2 Related work

Recently, great progress has been made in RL by approximating the policy and value functions through neural networks (Mnih et al., 2013; 2015; Schulman et al., 2015; Mnih et al., 2016). However, these RL methods often suffer from extremely slow training progress or fail to learn in sparse reward tasks. There have been several lines of approaches to tackling such problems, namely curriculum learning, hierarchical RL, intrinsic motivation and demonstration-augmented RL, each with its own constraints and assumptions.

Curriculum learning works by deconstructing a difficult, long-horizon learning task into a sequence of easy to hard tasks (Karpathy & Van De Panne, 2012). While classic curriculum is often defined manually, recently in RL a number of work has shown promising results automated curriculum generation (Pinto et al., 2017; Sukhbaatar et al., 2017; Wulfe & Hershey; Florensa et al., 2017b). Our work is closest to this line of work. In particular, reverse curriculum generation (Florensa et al., 2017a) has proposed a very similar curriculum generation through exploration backwards from the goal states, and has shown promising results in solving a variety of sparse reward problems. However, the critical difference of our approach from theirs is that we propose the use of the tendency reward trained discriminatively to summarize the past histories of successful and unsuccessful states, allowing our algorithm not to store the whole history of starting sets and further provide a dense shaping reward. Empirically our design choice is found to scale to larger state tasks with longer planning horizon better than this prior approach.

Hierarchical RL and intrinsic motivations are two other approaches for tackling the sparse reward problems. Hierarchical RL (Sutton et al., 1999; D.Kulkarni & Karthik Narasimhan, 2016) learns to plan using a hierarchical of low-level policies, often directly provided to the high-level planner, to drastically reduce the search dimension of the problem, while intrinsic motivation often provides additional dense rewards in terms of novelty (Stadie et al., 2015; Houthooft et al., 2016; Bellemare et al., 2016) or some notion of empowerment (Mohamed & Rezende, 2015) to improve the exploration under no rewards. However, both

approaches assume forward-based exploration to solve the sparse reward task, which becomes increasingly difficult as the dimensionality and the time horizon increase, unless strong structured prior is provided in HRL. Our method, with the similar assumptions as in reverse curriculum learning, allows much more efficient learning and exploration.

Lastly, there are a number of work that utilizes demonstrations to tackle such problems. Inverse reinforcement learning infers the underlying reward function from expert demonstrations(Ng et al., 2000; Abbeel & Ng, 2004; Ziebart et al., 2008), and recent work has used such reward function as potential-based shaping reward to accelerate learning (Brys et al., 2015; Harutyunyan et al., 2015). Another line of work directly made use of demonstrations to curriculize the training process (Popov et al., 2017). These methods, however, make strong assumptions about the quality of demonstrations they receive. By contrast, our optional checkpoint method only requires a small number of key states instead of several full demonstration trajectories, and is robust to noises. Specifically, for a challenging robotic manipulation task from perception, we only required 5 key states to solve the task.

3 Preliminaries and Assumptions

The definitions and symbols used in our model are listed in Appendix A

3.1 Preliminaries

We revisit the conventional reinforcement learning setting where an agent interacts with the environment Env sequentially. At each time step t, the agent receives an observation O_t and chooses an action a_t from a set of action space $\mathcal{A}$ according to its policy π. Then the environment will give the agent the next state s_{t+1} and a reward r_t. The goal of the agent is to maximize the expected return from each state s_t through accumulating the reward by $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$.

3.2 Assumptions for tendency reinforcement learning

The tendency reinforcement learning model is based on two assumptions that hold true for a variety of practical learning problems, especially for those learned in simulation.

Assumption 1. *When resetting, the agent can start from any state $s \in \mathcal{P}_i$.*

Assumption 2. *The interaction between agent and environment follows a Markov chain based on deterministic transformations, that is, $P(s_{i+1}|s_i, a_i) \in \{0, 1\}$.*

The first assumption agrees with learning in simulation, and can be satisfied in many real-world problems (e.g. robotic manipulations). If we could constrain the agent's start states to a valid phase, the agent would be able to train from these recommended start states, which proves to increase learning efficiency (Kakade & Langford, 2002). In case of long-term tasks where the phases extend with time, the number of possible start states might end up unnecessarily large. Under such circumstances, additional checkpoint related methods can also be applied to assist training, with which the assumption is fully compatible.

The second assumption indicates that the environment runs in a non-stochastic manner and always follows some fixed rules (e.g. laws of classical physics). Because of this, it makes sense to determine tendency rewards with a binary classifier, since there exists an absolute tendency for each state given a certain policy. In deterministic environments, irreversible processes may cause the phase extension to stuck on some absorbing states. However, our checkpoint scheme is able to solve this problem, details will be discussed in Appendix E.

4 The model

Naïve applications of conventional RL solver tend to fail in **goal-oriented tasks** with large state space. This kind of tasks are common in real world and can lead to an immense search

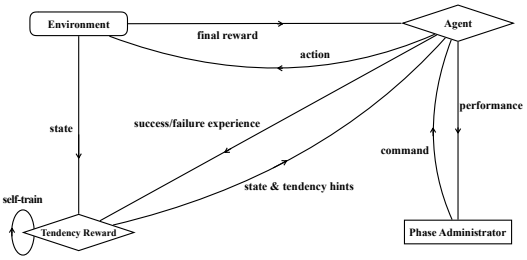


Figure 1: The schematic illustration of TRL.

space. Our Tendency RL is able to tackle this problem by reducing aimless searching with the application of 1) a novel tendency reward design that utilizes the agent's former experience; 2) a phase administrator that assigns customized training phases to the agent; 3) a powerful checkpoint scheme that shrinks search space and accelerates learning (optional).

Our Insight Our key idea is to let the agent utilize previous experiences through a tendency classifier trained on history trajectories in a supervised manner (Sec. 4.2). We define each phase as a set of start states. In training, phases are automatically generated by a phase administrator (Sec. 4.1) and gradually shift away from the final goal. In each phase, the agent learns from rewards given by the tendency classifier, and trains the classifier with its new experiences.

Algorithm 1: TendencyRL

input : Goal state s^g, starting phase $\mathcal{P}_1$, task start phase $\mathcal{P}^{task}$.
output: Trained policy π.

Phase index $i \leftarrow 1$;
if $\mathcal{P}_1 = \varnothing$ **then**
  $\mathcal{P}_1 \leftarrow SampleNearby(s^g, K)$;
Train policy π beginning from $\mathcal{P}_1$ with environmental reward;
Collect trajectories & train Tendency Reward $T(\cdot)$ (Sec. 4.2);
while $\mathcal{P}_i \cap \mathcal{P}^{task} = \varnothing$ **do**
  $\mathcal{P}_{i+1} \leftarrow PhaseAdministrator(\mathcal{P}_i, N_i)$ (Alg. 2);
  Train policy π beginning from $\mathcal{P}_{i+1}$ with hybird Reward (Eq. (2));
  Update Tendency Reward;
  $i \leftarrow i + 1$;
return π;

The input $\mathcal{P}_1 \neq \varnothing$ only when the optional checkpoint scheme is used, which will be discussed in Sec. 4.3.

4.1 Phase Administrator

Each time when the agent has made some progress in current phase (being able to accomplish the task starting from 80% of states in $\mathcal{P}_{current}$), the next phase $\mathcal{P}_{current+1}$ would be generated by the phase administrator based on the agent's performance in current phase.

Normally, during the ith phase extension process, we let the agent start from $\mathcal{P}_i$ and randomly take x_i steps. The newly reached states would then be included into $\mathcal{P}_{i+1}$ if the agent's success rate on that state is between a lower-bound α and an upper-bound β. The introduction of α and β helps to limit the new states to a proper difficulty (α is 0.1 and β is 0.9 in our experiments), and thus the agent is able to concentrate on exploiting new states and learn faster.

In each phase extension, a reasonable extension step size x_i is required. Generally, as x_i becomes larger, less phases are used to train the agent. However, the increased search space actually causes more training iterations. Therefore, in order to strike a balance, the phase administrator adjusts x_i according to the number of training iterations N_i in $\mathcal{P}_i$. This adjustment is achieved with a sigmoid function shown in equation (1).

$$x_i = \left\lceil x_{max} \times Sigmoid\left(\bar{N} - N_i\right) \right\rceil = \left\lceil \frac{x_{max}}{1 + e^{N_i - \bar{N}}} \right\rceil, \tag{1}$$

where $\bar{N}$ is the average of $N_1, N_2, \ldots, N_{i-1}$.

The procedure of the phase administrator is sketched in Algorithm 2.

Algorithm 2: Phase Administrator

input : Current phase $\mathcal{P}_i$, number of training iterations in the ith phase N_i.
output: The next phase $\mathcal{P}_{i+1}$.

Calculate x_i from N_i with Equation (1);
$\mathcal{P}_{i+1} \leftarrow \varnothing$;
while $|\mathcal{P}_{i+1}| < K$ **do**
  $s_{old} \leftarrow Sample\left(\mathcal{P}_i\right)$;
  $s_{new} \leftarrow RandomWalk\left(s_{old}, x_i\right)$;
  if $\alpha < SuccessRate\left(s_{new}\right) < \beta$ *and* $s_{new} \notin \mathcal{P}_{i+1}$ **then**
    $\mathcal{P}_{i+1} \leftarrow \mathcal{P}_{i+1} \cup \{s_{new}\}$;

4.2 Tendency Reward

The tendency reward in our model is a binary classifier (e.g. SVM, NN) that takes an observation O as input and outputs a judgment on whether that state leads to the final success. We name it tendency reward mainly because it provides slight guiding hints to speed up the training process in each phase and those large state space tasks thus become trainable.

In training, each time when the agent steps onto a state, it would refer to the tendency reward for a judgment. If the state is evaluated as "favorable", the agent would receive a positive tendency hint as a reward. Otherwise, a negative reward would be issued. To prevent the agent from stalling at a favorable state for unlimited tendency rewards, we defined that such revisited states do not provide any reward, and applied a Gaussian kernel (Appendix A.3) to detect repeated states.

The binary tendency classifier trains on both success and failure trajectories and we use logistic loss as the classifier loss. The definition of the hybrid reward is:

$$r(s) = \gamma T(s) + r_f(s), \tag{2}$$

where γ is a scale factor (10^{-3} in our experiments) based on the following hypothesis: a positive reward received from the environment should be more important than the output of $T(\cdot)$. Since the tendency hints are very subtle compared to the final reward, the agent's desire to achieve the final goal won't be interfered with.

4.3 Optional checkpoint scheme for extreme large state space task

In tasks with a vast state space, it is usually more difficult for $\mathcal{P}$ to reach any state s nearby $\mathcal{P}^{task}$. Those domains are difficult for both prior approaches and tendency RL. In those cases, we show that by providing just a few *checkpoints*, our tendency RL algorithm can take advantage of them to shrink the search space. Ideally, the checkpoints should be precise and indicate important states for the task, and in practice fine hand-engineering is often involved. However, with our tendency RL method, some rough key states would

be sufficient. The agent still extends its phases with a regular phase administrator, while after each extension, it additionally checks whether the newly generated phase has covered some states nearby a certain key state. If so, the next phase would be directly sampled from that key state rather than generated by the phase administrator. In this way, the search space can decrease remarkably. In experiments, our model also manifest robustness when encountered with misleading key states.

Given n as the number of checkpoints. The agent first extends new phases from s^g with the tendency RL algorithm described in Sec. 4. The extension would be ceased once the newly generated phase covers some states nearby s_{k_i}. The criteria of judgment is based on a Gaussian kernel approximation function (Appendix A.3) as is described in Eq. (3):

$$J(\mathcal{P}_{current}) = \sum_{s \in \mathcal{P}_{current}} \max_{t \in Sample(s_{k_i}, K)} k(s,t) \tag{3}$$

If J is larger than 30% of K, $\mathcal{P}_{current}$ is believed to cover some states around s_{k_i}. Then the next phase is sampled nearby s_{k_i} rather than extended from the current phase. After that, we apply the tendency RL algorithm again and extend new phases from it. The procedure will be repeated until $\mathcal{P}^{task}$ is reached.

Algorithm 3: TendencyRL with optional checkpoints

input : Goal state s^g, key states $s_{k_1}, \ldots, s_{k_n}$, task start phase $\mathcal{P}^{task}$.
output: Trained policy π for the extreme large space task.

for *j from* 1 *to* n **do**
  $\mathcal{Q}_j \leftarrow SampleNearby(s_{k_j}, K)$;
$\mathcal{R} \leftarrow \bigcup_{j=1}^{n} \{\mathcal{Q}_j\}$;
while $\mathcal{P}_{current} \cap \mathcal{P}^{task} = \varnothing$ **do**
  $\pi \leftarrow TendencyRL\left(s_g, \mathcal{P}_{current}, \bigcup_{\mathcal{Q} \in \mathcal{R}} \mathcal{Q}\right)$ (Alg. 1);
  for $\mathcal{Q}$ *in* $\mathcal{R}$ **do**
    if $\mathcal{Q} \cap \mathcal{P}_{current} \neq \varnothing$ **then**
      $\mathcal{R} \leftarrow \mathcal{R} \setminus \{\mathcal{Q}\}$;
      $\mathcal{P}_{current} \leftarrow \mathcal{Q}$;
      break;
return π;

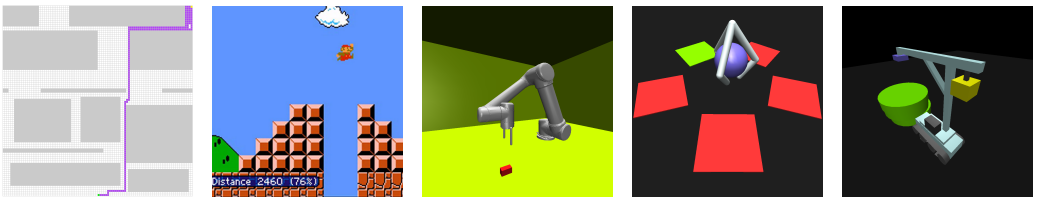
Note that our principle is distinct from conventional RL demonstration methods in that the agent is not eager to go after the checkpoints and does not get any additional reward from them. In fact, the small number of checkpoints are only used to shrink the search space, and our method also shows robustness to imperfect checkpoint settings in experiment where we deliberately mix misleading states into the checkpoints, and found that the policy learned by the agent doesn't get affected (Sec. 5.3).

5 EXPERIMENT

To evaluate our model, we design five experiments[1] with different levels of difficulties. The details are elaborated in Appendix B and C.

We first test the efficiency and validity of our model. After testifying the correctness of the overall model, we visualize the distribution of tendency hints and the starting sets of phases. Then we focus on the optional checkpoint scheme, verify its compatibility to our method and prove our method's robustness to checkpoints of different qualities. Finally,

[1]Demo videos of our experiments: https://sites.google.com/view/tendencyrl



(a) Maze (b) Super Mario (c) Grasping Task (d) Conveyance (e) Pick-and-place

Figure 2: Five environments on which we evaluate our model

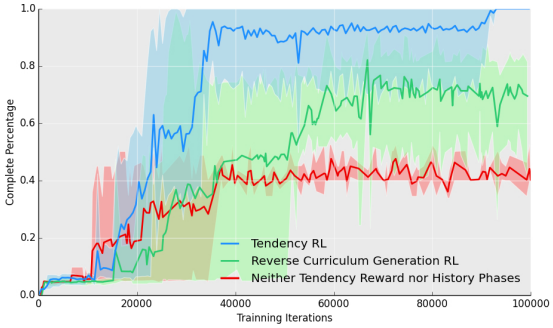


Figure 3: The training results of ablation experiments under three different conditions (mean to standard deviation with 5 random seeds).

we apply our model to long-term manipulation challenges with large state spaces to show our advantages.

5.1 Validity and efficiency of TRL

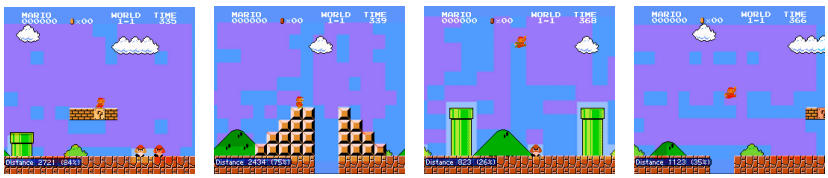
The training of the 40×40 maze task takes around 35000 training steps (1 hour and 40 minutes) with 6 threads on GPU. Then the agent can reach the exit from the start point of the maze with a 9×9 observation around it. To investigate the influence of tendency reward on training, we conduct an experiment comparing the training effect under three conditions: using tendency reward without history phases, using history phases without tendency reward and using neither tendency reward nor history phases. This is not only an ablation experiment but also a comparison with prior work, since the second condition is basically the reverse curriculum generation learning method (Florensa et al., 2017b). The result shows that our model performs well (Fig. 3).

5.2 Inspecting the learning results

The agent takes nearly 40000 training steps (around 21 hours) with 8 threads on GPU to learn how to complete the level 1-1 of Super Mario Bros. The policy learned by the agent is shown in Fig. 8. In order to find out the influence of the tendency reward, we traverse the whole states of some parts of the game and show both of the positive and negative tendency hints (Fig. 4).

As we can see in these figures, our agent has acquired the ability to find enemies nearby and dodge or even kill them by jumping up to their head. Besides, it also recognizes the danger of the pits and can choose the right time to jump over them. These figures clearly show the guiding effect of the tendency reward, which does a great help for the agent when it is far away from the final goal.

To figure out the phases generated by the phase administrator, we visualize the phases in Super Mario Bros, and we find such phases can effectively lead the agent to constantly learn



(a) (b) (c) (d)

Figure 4: The distribution of tendency hints in parts of Super Mario Bros game. The purple color represents positive hints while the other blue ones indicate negative hints.

Figure 5: This figure shows the some of the generated phases in training process. The points indicate the starting points of Mario. Points with same color belong to a same phase.

new states farther from the final goal. Parts of the phases are shown in Fig. 5. Different from the reverse curriculum method, our method neither needs to keep track of starting sets nor keeps restarting from all phases at each iteration, which is the key to scaling to long horizon tasks with large state space.

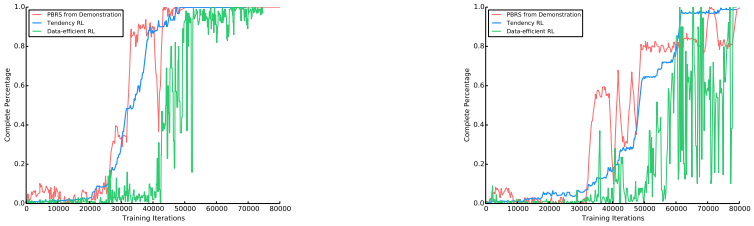
5.3 Compatibility and robustness to optional checkpoints

After testing and inspecting our normal tendency RL model, we then focus on the optional checkpoint scheme. We enlarge the maze to 100×100 also with a 9×9 observation around the agent. With 8 key states as the checkpoints, the training takes around 50000 training steps (7 hours and 14 minutes). We test our agent's robustness to imperfect checkpoints, related results can refer to Appendix D.3.

We also compare our model with two methods using demonstrations: data-efficient RL (Popov et al., 2017) and potential-based reward shaping (PBRS) from demonstration (Brys et al., 2015). The former resets the agent randomly to someplace on an expert trajectory while the latter uses an artificial potential reward shaped based on demonstrations. We test them with two kinds of checkpoints / demonstrations: "reasonable" and "unreasonable". As is shown in Fig. 6, data-efficient RL may fluctuate constantly when facing unreasonable demonstration. PBRS from demonstration requires a lot of human involvement to achieve high performance. Different from these two methods, our model only uses 10 evenly distributed key states on the trajectory as checkpoints and performs as good as PBRS from demonstration.

5.4 Success in large state space problems

The following experiments mainly focus on illustrating the practicality of our model on solving difficult, real-world robotic manipulation problems with minimal assumptions. When confronted with long-term challenges with large state space, to avoid wasting time on unnecessary states, our checkpoint scheme is applied to shrink the search space. As is described in Sec. 4.3, several roughly designed key states are sufficient to guide the agent through the large search space properly. In the three experiments, we only add less than 15 checkpoints to assist training (5 in robotic arm, 5 in conveyance challenge of the crane and 14 in pick-and-place task). The training process follows Alg. 3.



(a) Reasonable (b) Unreasonable

Figure 6: Learning curves of our model, the data-efficient RL and PBRS from demonstration with checkpoints / demonstrations of different qualities. PBRS from demonstration may outperform tendency RL with some hand-engineered demonstrations, while tendency RL is able to achieve a satisfactory performance only using crude checkpoints.

5.4.1 Grasping Task

The grasping task on robotic arm simulation is a typical large state space problem, where the gripper can move anywhere within the range of the arm. In our design, a camera is fastened to the forepart of the gripper and provides a 64×64 depth image as the agent's observation. We intuitively design 5 key states and train the whole task for 33360 iterations (4 hours 19 minutes) with 12 threads on GPU, and it turns out that our model can solve the task reliably.

To further investigate the relationship between training efficiency and scale of checkpoints, we conduct the same experiment with 5, 8, 11, 13 key points respectively, as is shown in Appendix D.4.

5.4.2 Conveyance Challenge

The conveyance challenge is also a 3D manipulation task with a larger state space. The agent should pick up the ball, take it to correct basket, and drop it right into that basket to receive a positive reward. The camera is also fixed on the gripper that produces 64×64 RGB images as input. Our model is trained for 99240 iterations (5 hours 2 minutes) with 12 threads on GPU.

Since this task involves multiple operations and has a large state space, we roughly add five key states to help the agent learn different skills, as is shown in Appendix D.5.

With the intention of finding out the distribution of "favorable" and "adverse" states given by the tendency reward, we make the agent travel through the whole environment at an appropriate height under two conditions: 1) the gripper jaw is open with the target ball fixed on the center of plane; 2) the ball is grasped by the gripper and moves with it. Then we record the tendency reward at each position (Fig. 7).

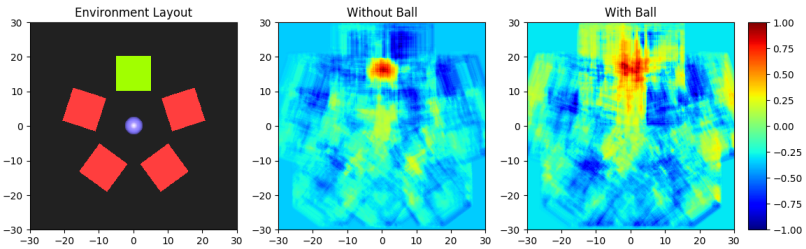


Figure 7: The tendency heat map at the height of 4cm, with different gripper poses in the conveyance task of a crane.

As expected, areas close to the target basket are assigned the highest tendency hints, which is quite natural as these states are reached before of after a successful drop. On the other hand, locations above those incorrect baskets tend to incur negative tendency hints. More details can be found in Appendix D.5.

5.4.3 RECURRENT PICK-AND-PLACE CHALLENGE

Of all the three experiments in robotic manipulation, this challenge has the largest state space due to endless combinations of the agent's location and orientation and enormous possible positions of plates. This time, the agent observes through a panorama camera that provides 160×32 first-person point of view.

The whole task can be divided into several identical pick-and-place operations that resemble the conveyance challenge. We provide 14 intuitively designed checkpoints to accelerate training (Appendix D.6). However, these checkpoints are not quite sufficient in such a huge state space, and the agent shows confusion from time to time.

Our phase administrator plays a significant role in phase arrangement, and actually makes up for the lack of expert data. Since the agent performs well within each pick-and-place operation, new phases are more concentrated on the intervals between operations.

6 DISCUSSION AND FUTURE WORK

We develop a tendency reinforcement learning algorithm that resolves complicated goal-oriented tasks, and evaluate it with multiple experiments based on raw perceptions. Our method consists of two core components: a binary tendency classifier that provides dense hints on whether a state leads to success and a phase administrator that generates a reverse curriculum. In five distinct environments, we find that tendency RL method is efficient and stable enough to converge on a reliable policy within hours even facing intractable problems, and does not require history data which is impossible to acquire in large space domains.

One limitation of our current approach is that our model requires the environment to be capable of resetting to some arbitrary states, which is also a limitation of the reverse curriculum generation method (Florensa et al., 2017b). A promising future direction is to additionally train reset controllers that automatically reset the system to the initial state following each episode, together with a corresponding forward controller (Han et al., 2015).

In future work, our goal is to apply the tendency RL method to real world robotic manipulation tasks with perceptions. Furthermore, we plan to extend the binary tendency classifier into a predictor that outputs the expected success rate on current state, so that our model can also be applied to stochastic environments. Additionally, we intend to release a subset of our code and the trained models to facilitate further research and comparisons.

REFERENCES

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1. ACM, 2004.

asfdfdfd. `https://github.com/asfdfdfd/fceux`, 2012.

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*, pp. 1471–1479, 2016.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E Taylor, and Ann Nowé. Reinforcement learning from demonstration through shaping. In *IJCAI*, pp. 3352–3358, 2015.

Tejas D.Kulkarni and Josh Tenenbaum Karthik Narasimhan, Ardavan Saeedi. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *nips*, 2016.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017a.

Carlos Florensa, David Held, Markus Wulfmeier, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300*, 2017b.

Weiqiao Han, Sergey Levine, and Pieter Abbeel. Learning compound multi-step controllers under unknown dynamics. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 6435–6442. IEEE, 2015.

Anna Harutyunyan, Tim Brys, Peter Vrancx, and Ann Nowé. Shaping mario with human advice. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1913–1914. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pp. 1109–1117, 2016.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Nineteenth International Conference on Machine Learning*, pp. 267–274, 2002.

Andrej Karpathy and Michiel Van De Panne. Curriculum learning for motor skills. *Advances in Artificial Intelligence*, pp. 325–330, 2012.

koltafrickenfer. `https://github.com/koltafrickenfer/gym-super-mario`, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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):529–533, 2015.

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*, pp. 1928–1937, 2016.

Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 2125–2133, 2015.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pp. 663–670, 2000.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. *arXiv preprint arXiv:1703.02702*, 2017.

Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1889–1897, 2015.

Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

Sainbayar Sukhbaatar, Ilya Kostrikov, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112 (1-2):181–211, 1999.

Blake Wulfe and David Hershey. Cs234 project final report: Approaches to hierarchical reinforcement learning.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A Definitions and formulae

A.1 Term definitions

- **Final goal**: States that indicate the task is successfully finished (e.g. the exit in maze).
- **Phase**: A set of start states at a certain time period.
- **Start state**: A state where the agent starts for the final goal in a certain phase.
- **Success state**: The states leading to the final goal by current policy.
- **Task start phase**: States where the entire task begins, which is the final aim of phases extensions (e.g. the start point in maze).
- **Checkpoint**: A small number of key states artificially designed to shrink the search space in large space tasks.

A.2 Symbol definitions

The symbols used in the tendency reinforcement learning model are described in table 1.

Table 1: List of symbols

Symbol	Definition
Env	Environment that the agent interacts with.
$\mathcal{P}_i$	The set of start states in i^{th} phase.
$\mathcal{P}^{task}$	The task start phase.
K	Number of states in each phase.
O	Observation acquired from the environment.
s	State that the agent arrives at.
s^g	State where the agent reaches the goal.
r^f	Reward given by environment when the final goal is achieved.
π	Policy pursued by the agent.
$T(\cdot)$	Function of tendency reward.
$r(\cdot)$	Hybrid reward function consisting of $T(\cdot)$ and r^f
α, β	Lower and upper bound of success rate in phases extension.
N_i	Number of training iterations in $\mathcal{P}_i$.
x_i	Extension step size after the training of $\mathcal{P}_i$.
x_{max}	Max random walk steps in extension.
d	Number of recent frames used in Gaussian kernel.

A.3 Gaussian Kernel

A Gaussian kernel (Eq. (4)) is used in our experiments to detect repeated states. The most d recent frames are used in the kernel, as is shown in Eq. (5):

$$k(x, y) = exp\Big(- \sum_j \min(\max((x_j - y_j)^2 - \delta, 0), 1)/\sigma\Big) \tag{4}$$

$$\rho(s^{now}) = \max_{1 \le i \le d} k(s^{now}, s^i), \tag{5}$$

We assume that s^{now} is a revisited state if ρ is larger than 0.9. The parameter d is 20% of the max steps of each environment and $\sigma = 1$ & $\delta = 0$ are constants in our experiments.

B Environment Details

B.1 Maze

Our first environment is a maze task. The size of the whole maze is 100×100. There is only one exit (final goal) in the middle of the bottom line ($[100, 50]$) and one starting point at $[0, 50]$. The agent is required to find its own way to move from the starting point to exit with a limit observation covering a 9×9 area around it. At each time-step, the agent has four actions, namely *walk north*, *walk south*, *walk east* and *walk west*, expressed as (N, S, E, W), and transitions are made deterministically to an adjacent cell, unless there is a block, in which case no movement occurs. The environment only gives 1 point reward when the agent has accomplished the finial goal. On other conditions, the reward remains 0.

B.2 Super Mario Bros

The second environment is level 1-1 of Super Mario Bros (Brockman et al., 2016; koltafrickenfer, 2017; asfdfdfd, 2012). The agent should complete the level from start point via a 16×13 observation, which indicates the nearby situation. When the agent reaches its destination, it would receive 1 point reward, otherwise it receives 0 point. The agent has 5 actions: *jump*, *right*, *left*, *jump to right* and *jump to left*, expressed as (J, R, L, JR, JL). Mario could not achieve any reward by killing enemies or collecting coins.

B.3 GRASPING TASK

The third environment is a grasping task on a robotic arm simulation in MuJoCo. The end effector of the 6-DOF robotic arm is a two-finger gripper with a camera attached between the fingers, which produces 64×64 overlooking depth images as the agent's observations. In order to manipulate in a more intuitive fashion, poses of joints have been mapped into 3D coordinates, enabling the gripper to move in 6 different directions. There is also a special grasping action that closes the gripper jaws, by which the target cube can be grasped. The agent is initialized randomly above a plane, and 1 point reward would be received if and only if the target cube placed on the plane is captured by the gripper.

B.4 CONVEYANCE CHALLENGE

The fourth task is a conveyance challenge of a crane simulated by the physics engine MuJoCo. The agent can interact with a ball placed on a plane using a 3-finger gripper connected to the crane, with 8 actions available (6 for free movements and 2 for gripper manipulations). Surrounding the ball from a distance are some target baskets, among which one basket is randomly selected and colored green. The agent is able to observe through a camera fixed on the gripper that produces 64×64 RGB images, and only by dropping the ball into the green basket would the agent receive 1 point reward.

B.5 RECURRENT PICK-AND-PLACE

The fifth task involves pick-and-place operations simulated in MuJoCo, which requires the agent to stack some plates into a neat pile. This time, the agent observes through a panorama camera that provides 160×32 first-person point of view. Fixed before the camera is a electromagnet capable of attracting one plate at a time, which is used for plate transfer. Available actions include *turning left / right*, *stepping forward / backward*, *charging / releasing the electromagnet*. The final goal is to gather and pile up all the plates into a straight stack, which is the only approach for the agent to obtain 1 point reward.

C NETWORK ARCHITECTURES

C.1 MAZE

The convolution neural network of the policy network has a first layer with 12 4×4 filters of stride 1, followed by a layer with 24 3×3 filters of stride 1, followed by a layer with 48 2×2 filters of stride 1. Then we add a LSTM layer with 384 units which followed by a fully connected layer with 96 hidden units.

C.2 SUPER MARIO BROS

The convolution neural network of the policy network has a first layer with 48 4×4 filters of stride 1, followed by a layer with 48 3×3 filters of stride 1, followed by a layer with 96 2×2 filters of stride 1. Then we add a LSTM layer with 960 units which followed by a fully connected layer with 240 hidden units.

C.3 EXPERIMENTS WITH MUJOCO ENGINE

The convolution neural network of the policy network has a first layer with 12 4×4 filters of stride 1, followed by a layer with 24 3×3 filters of stride 1, followed by a layer with 48 2×2 filters of stride 1, followed by a layer with 48 2×2 filters of stride 1. Then we add a LSTM layer with 1536 units which followed by a fully connected layer with 96 hidden units. Especially, in the pick-and-place challenge, there is an additional convolution layer with 96 2×2 filters of stride 1, and the LSTM layer contains 480 units.

D SUPPLEMENTARY MATERIALS FOR EXPERIMENTS

D.1 SUPER MARIO BROS DETAILS

D.1.1 TRAINING RESULT

The training result of Super Mario Bros is shown in Fig. 8.

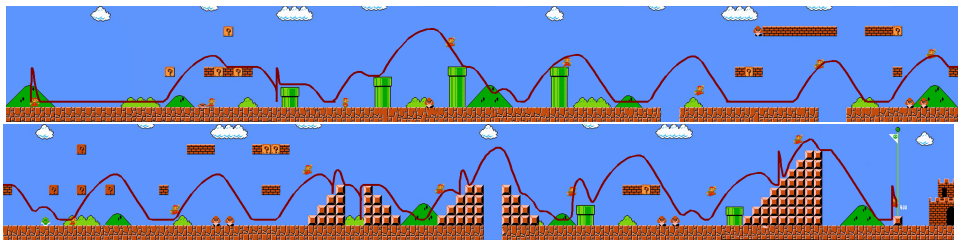


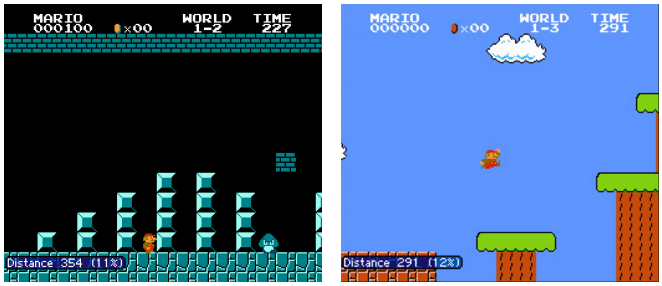
Figure 8: The training result of the Super Mario Bros level 1-1. The red curve indicates the Mario's policy.

D.1.2 COMPARISON WITH CURIOSITY-DRIVEN METHOD

In Super Mario Bros, we compare our model with another one trained based on curiosity-driven (Pathak et al., 2017). We let the agents randomly move several steps at the beginning before running the models, then compare the completion degree [2] of two models. With the environment only providing a reward when Mario reaches the final goal, we find that our method obtains absolute advantage (Table 2). We also use the model trained in level 1-1 to run the level 1-2 and 1-3 of the game. Since the latter two levels are not trained and require new skills that the first level does not include (Fig. 9), both of the methods may suffer more failures.

Table 2: Compare with the model trained based on curiosity-driven

Model	**Completion degree in level 1-1**(%)	**Completion degree in level 1-2**(%)	**Completion degree in level 1-3**(%)
ours	100	22	17
curiosity-driven	28	1	12



(a) Level 1-2 (b) Level 1-3

Figure 9: The cases where the agent needs new skills in level 1-2 and level 1-3.

[2] The mean length they move divided by all length of current level

D.2 COMPARISON WITH MAXIMUM ENTROPY IRL

We try the maximum entropy IRL (Ziebart et al., 2008) in the 100×100 maze task with more than two days' training and about 44G virtual memory used, but it still failed to form a feasible reward function. Since there are 10,000 states in the maze, the traditional IRL methods seem to suffer from the bottleneck of computing ability.

D.3 ROBUSTNESS TO IMPERFECT CHECKPOINTS

To test our model's robustness to imperfect checkpoint settings, instead of arranging the checkpoints onto the optimal trajectory, we set some of them onto a byway. We choose 8 special phases sampled from the misleading checkpoints and show the policy the agent has learned in these phases. As is shown in Fig. 10, in the sixth phase, the agent refuses to detour from the left side and finds a shortcut direct to the exit. A possible explanation is that our tendency reward only provide slight guiding hints which do not bother the agent's exploration for new policies.

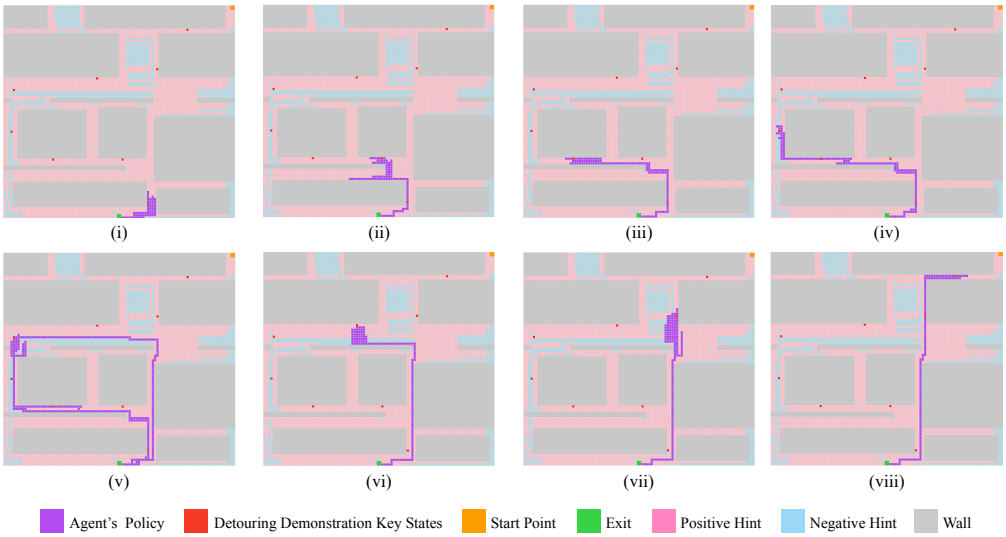


Figure 10: The agent's policies in 8 special phases sampled from misleading key states. The purple points are collected by combining the testing results for 30 times. In the end the detouring checkpoints are completely neglected by the agent.

D.4 THE INFLUENCE OF CHECKPOINT SCALE ON TRAINING EFFICIENCY

We conduct the grasping experiment with 5, 8, 11, 13 key points respectively. The influence of checkpoint scale on training efficiency can refer to Fig. 11.

D.5 CONVEYANCE CHALLENGE DETAILS

An interesting observation in conveyance challenge is, the areas with positive tendency hints are more concentrated and the boundary between "favorable" and "adverse" states is more explicit when the gripper jaw is open. This is because the states where the agent is not holding the ball are often related to grasping and dropping, which require more accuracy than just transferring the ball.

This experiment also proves that our agent has superb robustness to imperfect checkpoint settings. In our previous assumptions, the agent should rise to a proper height where all baskets were completely in view, before moving towards the correct basket (Fig. 12). However, it turns out that the agent learns to determine the target basket location at a lower height, where only portions of the baskets are in view.

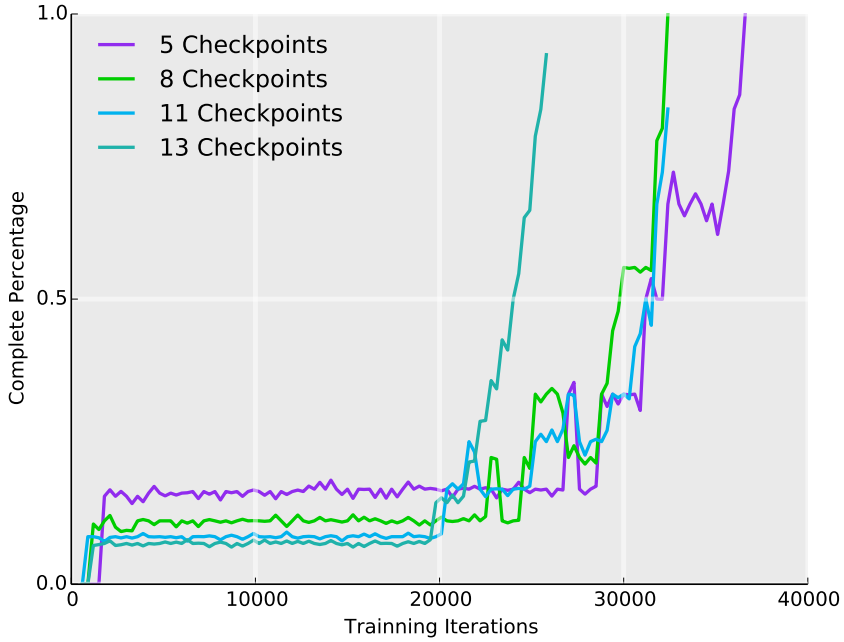
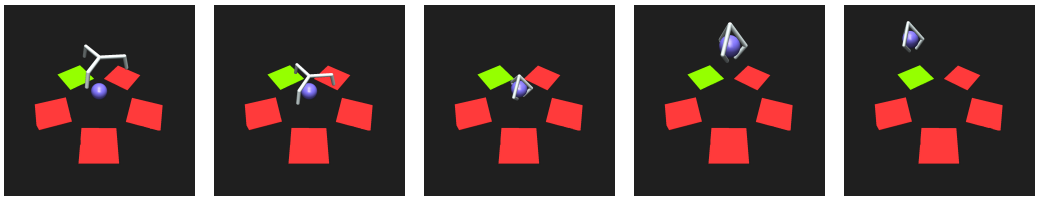


Figure 11: Learning curve of the grasping task with different numbers of checkpoints. The training efficiency is relatively higher when 8 checkpoints are used. If the number of checkpoints is too large or too small, training tends to become less efficient, but still stable.



(a) Diving (b) Grasping (c) Rising (d) Gliding (e) Dropping

Figure 12: Five key states for the conveyance challenge and skills required by these states.

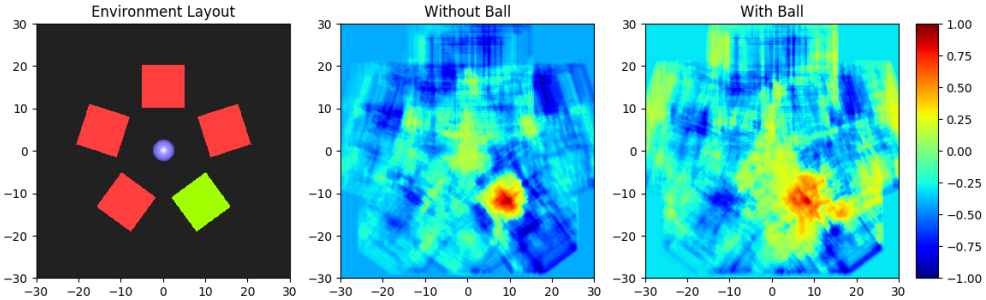


Figure 13: Another tendency heat map of the conveyance task.

D.6 PICK-AND-PLACE DETAILS

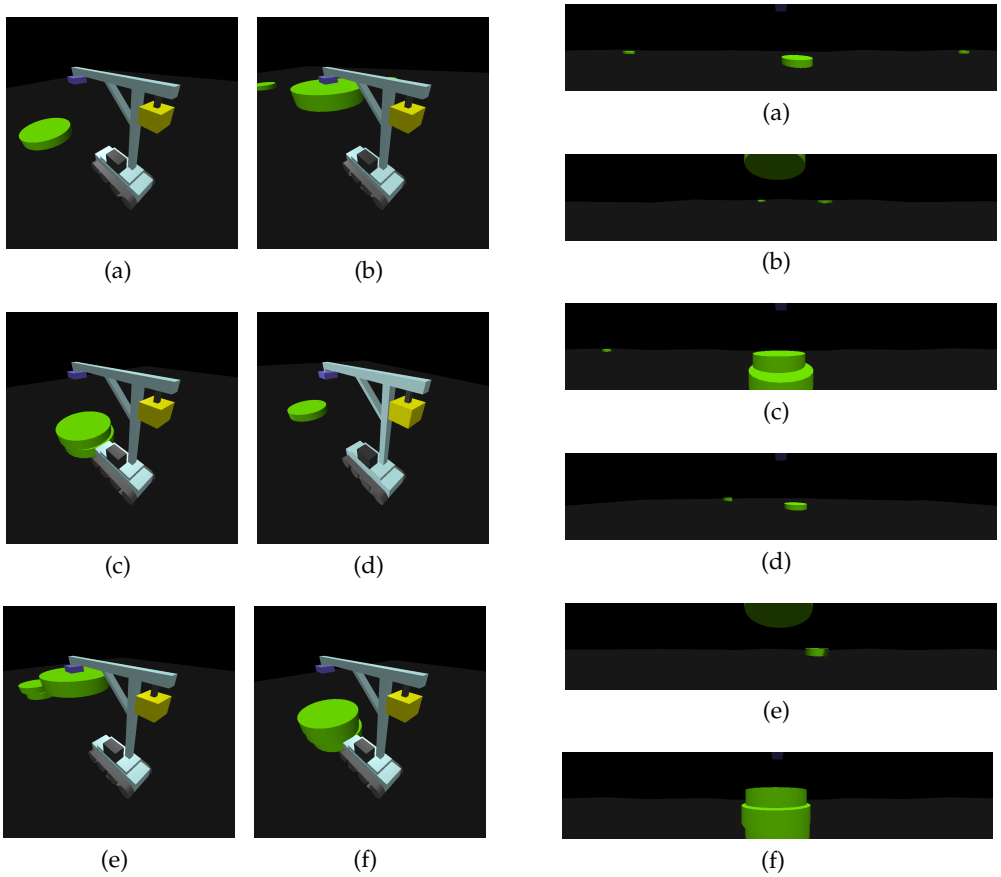


Figure 14: Some key states and the corresponding observations in the pick-and-place challenge.

E OTHER DISCUSSIONS

The regular reverse curriculum algorithm usually fails if there exists an **irreversible process** in the system. The irreversible process is defined as:

$$\exists s, s^{'} \in \mathcal{S}: \ (\exists n > 0: \ P(s_n = s'|s_0 = s) > 0) \wedge (\forall n > 0: \ P(s_n = s|s_0 = s') = 0) \quad (6)$$

In such cases, the states s and s' are not connected, and an agent starting from s' will never reach s since that probability is 0. We define a **absorbing state** s_a as a state that satisfies

$$P(s_a|s_a) = 1 \wedge \forall s \in \mathcal{S}: \ s \neq s_a \longrightarrow P(a|s) = 0 \quad (7)$$

To be more generalized, we define a set $\mathcal{S}_a$ of states to be a **absorbing set** if it satisfies

$$P(s'|s) = 0 \text{ if } s \in \mathcal{S}_a \wedge s' \notin \mathcal{S}_a \quad (8)$$

Consider a phase extension progress where $\mathcal{P}_{i+1}$ is generated from $\mathcal{P}_i$, if a large portion of states in $\mathcal{P}_{i+1}$ belong to some absorbing sets, it would be hard to for the new phase to include elements not in these absorbing sets. Therefore, the training is likely to be contained

within these sets and no actual progress could be made since the phase extension makes no sense.

However, with additional checkpoints, this limitation could be avoided even with irreversible processes and absorbing sets. The mechanism is described as follows: When we sample states nearby a checkpoint that is not in any absorbing set, the sampled states might happen to belong to some absorbing set. Although phase extensions are constrained within the absorbing sets, the generated phase might also cover some states sampled nearby a checkpoint. Thus, according to the phase administrator algorithm (Alg. 2), that checkpoint could still be reached.