Automatic Curricula via Expert Demonstrations

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose Automatic Curricula via Expert Demonstrations (ACED), a reinforcement learning (RL) approach that combines the ideas of imitation learning and curriculum learning in order to solve challenging robotic manipulation tasks with sparse reward functions. Curriculum learning solves complicated RL tasks by introducing a sequence of auxiliary tasks with increasing difficulty, yet how to automatically design effective and generalizable curricula remains a challenging research problem. ACED extracts curricula from a small amount of expert demonstration trajectories by dividing demonstrations into sections and initializing training episodes to states sampled from different sections of demonstrations. Through moving the reset states from the end to the beginning of demonstrations as the learning agent improves its performance, ACED not only learns challenging manipulation tasks with unseen initializations and goals, but also discovers novel solutions that are distinct from the demonstrations. In addition, ACED can be naturally combined with other imitation learning methods to utilize expert demonstrations in a more efficient manner, and we show that a combination of ACED with behavior cloning allows pick-and-place tasks to be learned with as few as 1 demonstration and block stacking tasks to be learned with 20 demonstrations.

1 Introduction

Imagine a robot factory is trying to manufacture home support robots and sell them all over the world. When the customer asks the robot to complete a novel task, he will likely not be willing to program a detailed task specification or carefully design a set of rewards to guide the robot. What the customer is more likely to provide is probably a high-level goal, a handful of demonstrations, or a combination of both. Classical task and motion planning solutions to robotic manipulation [1] often require carefully engineered domain specifications, but it is infeasible to pre-define all possible tasks the robot might be asked to do in all possible environments. Reinforcement learning (RL) approaches don't require the domain model, though they typically only work well in well-structured environments with carefully designed dense reward signals [2]. Solving RL problems with only sparse or binary rewards has been a long-standing challenge for researchers and many approaches have been proposed, including intrinsic motivation [3; 4; 5], hierarchical RL [6; 7] and curriculum learning [8; 9]. On the other hand, imitation learning [10; 11] methods resort to expert demonstrations instead of hand-designed reward signals and have shown impressive performance, especially in tasks where the reward functions are tricky to define but demonstrations are easier to obtain. In many tasks in robotics, both a binary reward for tasks success and a small amount of demonstrations can be provided easily, so can we use demonstrations to overcome the challenging exploration problem RL agents face in these long-horizon sparse-reward tasks?

An intuitive idea of utilizing human demonstrations to overcome exploration challenges is to automatically generate a curriculum through demonstration trajectories. With a well-designed curriculum, RL agents can first solve simpler problems where rewards are easy to obtain in order to master the skills

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

that can increase their chance of getting rewards in the challenging tasks. In many manipulation tasks, designing curricula can be tricky and tedious, but human demonstrations can naturally be converted into curricula. Suppose we have a demonstration trajectory $\tau = (s_0, s_1, \ldots, s_{T-1}, s_T)$, where s_0 is the initial state and s_T is the goal state. If we assume the demonstration trajectory solves the task in a reasonable way without deliberate detours (even though it could be suboptimal), then it is also reasonable to assume that s_{T-1} is closer to s_T in the task space than s_1, which means an RL agent starting from s_{T-1} will likely have a higher chance of reaching s_T than an agent starting from s_1 within a limited time of random exploration. Therefore, among all tasks with binary rewards that are only given when the goal s_T is fully reached, the ones with agents initialized at s_{T-1} should be easier than the ones with agents initialized at s_1. We hypothesize that agents who have already learned how to reach s_T from s_{T-1} can provide a warm start for agents trying to reach s_T from s_1, and that these tasks starting from different initial states can form a systematic curriculum for learning challenging long-horizon tasks with sparse rewards.

With this intuition, we propose Automatic Curricula via Expert Demonstrations (ACED), a RL approach which uses states from different sections along demonstration trajectories as reset states and controls the curriculum by moving reset states from the end of the demonstrations to the beginning based on the agent's performance. Although prior works [12; 13] evaluated similar ideas in grid world environments and games, the ability of utilizing an arbitrary number of demonstrations and generalizing to random unseen initializations and goals was not provided. In this paper, we evaluate ACED in robotics pick-and-place tasks and block stacking tasks with only binary rewards, two challenging tasks in the continuous control domain that haven't been solved by vanilla RL algorithms, and analyze the influence of the number of demonstrations and the total number of sections the demonstrations are divided into on ACED's performance. An additional advantage of ACED is that it can be naturally combined with many other methods of utilizing human demonstrations in order to further improve its performance or reduce the number of demonstration trajectories needed, and a combination of ACED and behavior cloning (BC) is demonstrated as an example in this paper. Empirical results show that pick-and-place can be learned with as few as 1 demonstration, and block stacking can be learned with as few as 20 demonstrations.

2 Related Work

2.1 Curriculum Learning

Curriculum learning [8] is a continual learning method that accelerates the learning progress by gradually increasing the task difficulty. It has seen success in many applications including language modeling [14], autonomous navigation [15; 9] and robotic manipulation [16]. However, many curriculum-based methods only involve a small and discrete set of manually generated task sequences as the curriculum, and existing automated curriculum generating methods often assume prior knowledge on how to manipulate the environment [17], or inherit the instability of adversarial methods and bias the exploration to a small subset of the tasks [18; 19]. [16] introduced the idea of automatically generating initial states closer to the goal state in order to speed up training, but inevitably face the challenge of infeasible randomly-generated initial states and can't be trivially extended to problems where the action space distance is not a good indicator of task difficulty. In order to address these issues, we propose to use states from expert demonstrations as initial states to guarantee feasibility and provide more accurate indication of task difficulty.

2.2 Learning from Demonstration

Learning from demonstration (LfD) is widely used in tasks where the reward function is hard to define but demonstrations are relatively easier to obtain. Behavior cloning [20; 21] is a classical LfD approach that utilizes supervised-learning to train agents that imitate demonstration behaviors. Although BC has seen success in various fields including autonomous driving [22; 23] and robotics manipulation [24], it inevitably demands a large amount of demonstrations and its performance often suffers from data distribution mismatch [25]. Inverse reinforcement learning [26; 27] infers the reward function through demonstrations in order to avoid manual reward engineering, but it is fundamentally challenging due to its ambiguity in solutions since one trajectory can often be explained by many different reward functions [28]. LfD approaches based on Generative Adversarial Networks (GAN) [10; 29; 11] have effectively scaled to applications with relatively high-dimensional

environments, but challenges due to unstable GAN training have significantly restricted their success in long-horizon tasks with complicated environments. Another popular approach to effectively utilize expert demonstrations is to combine LfD with RL [30; 31; 32]. The advantage of ACED is that it can easily be combined with many existing LfD methods for more efficient utilization of expert demonstrations, including BC, GAN-based methods [10] and methods that add demonstrations in RL replay buffers [30; 32]. In the empirical evaluation section in this paper, we demonstrate the performance of our approach when combined with BC and show that it provides better convergence performance compared to using ACED only.

2.3 State Resetting

State resetting is widely used in RL for introducing expert knowledge, applying curricula or providing safety guarantees. [33] and [32] reset some training episodes to states from expert demonstrations to simplify the exploration challenges in long-horizon tasks. [34] and [35] show that learning both the forward policy and the reset policy can not only reduce human effort in real-world robotics training but also accelerate training by automatically forming a curriculum. [36] introduces "teacher's interventions" via state resetting to avoid costly mistakes during learning in safety-critical applications. Similar to our proposed approach, [12] and [13] reset the initial states during training to demonstration states in order to form a sequence of curricula. However, [12] pointed out its limitations in terms of generalizing to unseen states and didn't provide evaluation on continuous control tasks or in-depth analysis on different component's influence on the overall performance, whereas [13] used a set of fixed rules for switching curricula instead of adapting it based on the agent's performance. In contrast to [12] and [13] which can only utilize one demonstration trajectory, ACED allows for arbitrary numbers of demonstrations through trajectory sectioning. In this paper, we evaluate ACED on continuous control tasks and analyze the influence of the number of demonstration trajectories and the number of sections they are divided into on ACED's overall performance.

3 Preliminaries

3.1 Reinforcement Learning

The problem studied in this paper is formulated as a Markov Decision Process (MDP) defined by states $\mathbf{s} \in \mathcal{S}$, actions $\mathbf{a} \in \mathcal{A}$, a transition model $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, and a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. $\mathcal{S}$ and $\mathcal{A}$ represent the state space and the action space respectively. The objective of the RL problem is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes $J = \mathbb{E}_\pi[\sum_\tau r(\mathbf{s}_t, \mathbf{a}_t)|\mathbf{a}_t \sim \pi(\mathbf{s}_t), \mathbf{s}_0 \sim p_0(\mathbf{s})]$, where τ denotes the rollout trajectory [37]. ACED can work with any standard RL algorithm, and in this paper we demonstrate its performance using Proximal Policy Optimization (PPO) [38] algorithm and Deep Deterministic Policy Gradient (DDPG) [39].

3.2 Behavior Cloning

Given expert demonstration trajectories $\mathcal{T} = \{\tau_1, \ldots, \tau_N\}$ where each trajectory includes state-action pairs, i.e. $\tau_e = (\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_t, \mathbf{a}_t, \ldots, \mathbf{s}_T)$, the objective of BC is to learn a mapping from states to actions through supervised learning in order to imitate expert behaviors. Due to BC's demand for a large amount of demonstrations and its poor generalization performance in unseen states, it is often used to pre-train the policy network for other imitation learning or RL approaches as a warm start instead of as a standalone imitation learning approach. The ACED method in this paper can also be combined with BC by using it to pre-train policy networks, and we present this combination in Section 4. We compare the performances of ACED with or without BC in Section 5. Note that demonstration trajectories with state-action pairs are only required by BC, and if ACED is used without BC, then demonstration trajectories with only states are sufficient.

4 Approach: Automatic Curricula via Expert Demonstrations (ACED)

In order to solve long-horizon manipulation tasks with binary rewards, ACED constructs a curriculum by sampling states from expert demonstration trajectories as initializations for each training episode, where the samples initially come from near the end of the demonstration trajectories and gradually move forward as the agent improves its performance.

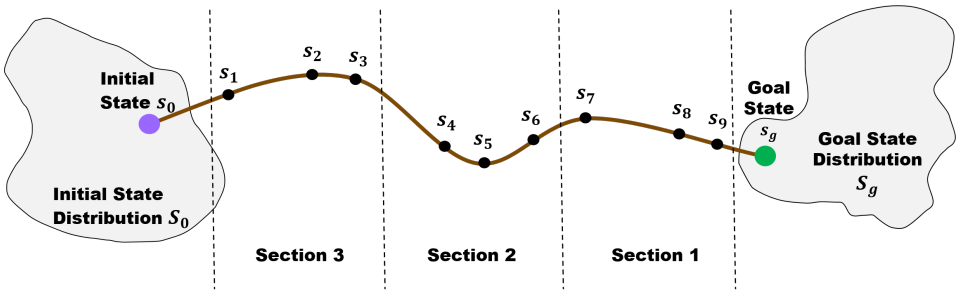


Figure 1: Example of demonstration trajectory segmentation: an expert demonstration trajectory can be divided into sections where larger section number indicates being closer to the initial state. The total number of sections is also called the total number of curricula C_{max}, and in this example $C_{max} = 3$. Normal rollout workers randomly sample an initial state from the initial state distribution S_0 and a goal state from the goal state distribution S_g. For curriculum rollout workers, the environments are reset based on the curriculum number C: curriculum-C tasks reset the environment to a section-C state on a randomly selected demonstration trajectory. ACED starts training with $C = 1$ and gradually moves reset states towards the beginning of demonstration trajectories by increasing C. When ACED switches to normal rollout workers, the reset states are drawn from the actual S_0 the target task specifies, and this is when it starts to generalize to unseen initializations.

Algorithm 1: Automatic Curricula via Expert Demonstrations

Input:
T: number of iterations
C_{max}: total number of curricula
$\mathcal{T} = \{\tau_1, \ldots, \tau_N\}$: demonstration trajectories
ϕ: curriculum switching threshold for average return
t: period for checking average return
n: number of episodes used to compute average return
Output:
π: policy

```
Initialize policy parameters with Behavior Cloning Algorithm
Initialize curriculum number C ← 1
Initialize rollout worker W ← CurriculumRolloutWorker
Initialize rollout trajectory buffer E ← {}
for i = 1, 2, ..., T do
    τ ← W.rollout(C, C_max, T, π)
    Add τ to E
    Send E to RL Algorithm and update π
    if i mod t == 0 then
        R ← Evaluate the average return on the most recent n episodes
        if R ⩾ φ then
            if C < C_max then
                C ← C + 1
            else
                W ← NormalRolloutWorker
```

We first collect a set of expert demonstration trajectories $\mathcal{T}$ and represent each trajectory $\tau_e \in \mathcal{T}$ as a discrete sequence of states at each time step: $\tau_e = (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{T-1}, \mathbf{s}_T)$, where the initial state $\mathbf{s}_0$ is randomly sampled from the initial state distribution S_0 and the final state $\mathbf{s}_T$ has a probability of $p_{success}$ to reach a goal state randomly sampled from the goal distribution S_g, i.e. $\mathbb{P}(\mathbf{s}_T = \mathbf{s}_g) = p_{success}, \mathbf{s}_g \in S_g$. We refer to $p_{success}$ as the expert success rate. Each trajectory τ_e is then evenly divided into C_{max} sections, where section-C_{max} denotes the section at the beginning of τ_e (near $\mathbf{s}_0$) and section-1 denotes the one at the end (near $\mathbf{s}_T$). C_{max} is a hyperparameter referred to as the *total number of curricula*. Figure 1 provides an illustration of an example demonstration trajectory and its segmentation. A key assumption made in ACED is that all expert demonstrations are reasonable solutions to the problem and don't contain unnecessary detours, which guarantees that two states that are close in the demonstration trajectory are also close in the task space.

Algorithm 2: CurriculumRolloutWorker

Input:
C: current curriculum number
C_{max}: total number of curricula
$\mathcal{T} = \{\tau_1, \ldots, \tau_N\}$: demonstration trajectories
π: current policy
Output:
τ: rollout trajectory

1 Randomly select a trajectory from demonstrations $\tau_e \in \mathcal{T}$
2 $num_transitions = \text{len}(\tau_e) - 1$ `/* Make sure to not sample the goal state */`
3 $interval = \text{RoundDown}(num_transitions/C_{max})$ `/* Divide` τ_e `into` C_{max} `intervals */`
4 $index = \text{RandInt}(interval) + interval \times (C_{max} - C)$ `/* Randomly select a state from the segment of` τ_e `corresponding to the current` C `*/`
5 $\mathbf{s}_{init} = \tau_e[index]$
6 $\tau = \text{Rollout}(env, \mathbf{s}_{init}, \pi)$

Algorithm 1 describes the overall framework of ACED. Given the expert demonstration set $\mathcal{T} = \{\tau_1, \ldots, \tau_N\}$, we can pretrain the policy network with BC (line 1). We refer to this version of the algorithm as ACED with BC, and if Algorithm 1 is executed without line 1, then we call it ACED without BC. The performances of the two versions are compared in pick-and-place tasks in Section 5.1. We use a set of parallel environments to generate rollout data for RL training. We refer to the original environment that resets to initial states sampled from S_0 as the *normal rollout worker*, and the curriculum-based environment that resets to demonstration states the *curriculum rollout worker*. Details of the curriculum rollout worker is described in Algorithm 2. At the beginning of training, all parallel environments are set to be curriculum rollout workers (as shown in Algorithm 1 line 3), and the switch from curriculum rollout workers to normal rollout workers are controlled by *curriculum number* $C \in \{1, 2, \ldots, C_{max}\}$. At each iteration, Algorithm 1 will collect training data using rollout workers and optimize the policy using the RL algorithm of choice (line 6 - 8). C is initialized to be 1, and every t iterations the algorithm will check the average return from the most recent n episodes and compare it with a threshold ϕ (line 9 - 15). When the average return exceeds the threshold ϕ, we add 1 to the current curriculum number C if it hasn't reached C_{max}, or switch to the normal rollout worker if C has reached C_{max}.

At each rollout, as shown in Algorithm 2, the curriculum rollout worker will first randomly select a demonstration trajectory τ_e and divide it into C_{max} sections with equal number of states (line 3). Based on the current curriculum number C, the curriculum rollout worker resets the environment to a randomly selected state from section-C on the demonstration trajectory τ_e (line 4 and 5). It will then rollout a trajectory using the current policy and return it to Algorithm 1. In our implementation, each rollout worker keeps track of its own curriculum number and the switch from a curriculum rollout worker to a normal rollout worker is independent from other parallel workers' C value.

5 Empirical Evaluation

We evaluate our approach on two tasks in the Fetch environment in OpenAI Gym [40]: a pick-and-place task and a block stacking task. The pick-and-place task is adapted from Gym directly and the block stacking task is adapted from [41]. The goal of the pick-and-place task is to move a block randomly placed on the tabletop to a goal pose that could be either in the air or on the tabletop, and the goal of the stacking task is to move two randomly placed blocks to their corresponding goal pose where the yellow block is stacked on top of the green block on the tabletop. The majority of results presented in this section use PPO as the RL algorithm, and we demonstrate ACED's off-policy performance with DDPG in the pick-and-place task. The demonstration trajectories are generated from a hand-coded straight-line policy, i.e. the robot is instructed to follow a straight-line trajectory to reach the object, grasp the object and then follow a straight-line trajectory to reach the goal. For both tasks, we use a binary reward function, where $r = 1$ indicates all blocks are within the distance threshold to their corresponding goal poses and $r = 0$ otherwise. The episode is terminated immediately if $r = 1$ is obtained even if the maximum episode length hasn't been reached. Additional implementation details are presented in Appendix A. In the results presented in this section, we

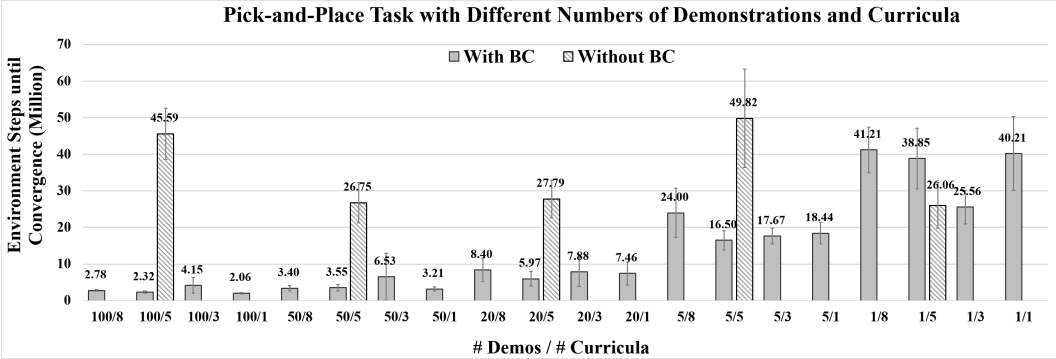


Figure 2: Number of environment steps ACED with BC and ACED without BC take to train pick-and-place tasks with PPO until convergence with different values of the number of demonstration trajectories $|\mathcal{T}|$ and the total number of curricula C_{max}. The bars represent the mean of 10 runs with different random seeds and the error bars represent the 90% confidence interval.

mainly focus on the convergence performance and the success rate, but selected examples of the learning curves in the pick-and-place environment are shown in Appendix B.

5.1 Pick-and-Place Tasks

In this section, we compare the performance of ACED with BC and ACED without BC on pick-and-place tasks in the Fetch environment. In order to study the influence of the number of demonstration trajectories $|\mathcal{T}|$ and the total number of curricula C_{max}, we test ACED with BC with $|\mathcal{T}| = 100, 50, 20, 5, 1$ and $C_{max} = 8, 5, 3, 1$. ACED without BC is only tested with $|\mathcal{T}| = 100, 50, 20, 5, 1$ and $C_{max} = 5$, since $C_{max} = 5$ is the best performer in ACED with BC experiments in terms of convergence speed. Here we define convergence as having a training success rate of stably above 90% and being able to accomplish most tasks during test time. We compare their convergence performance during training with PPO in Figure 2 and their success rate performance during testing in Table 1. We also tested ACED with DDPG with $|\mathcal{T}| = 5$ and $C_{max} = 5$, and the average steps to convergence is 5.07 million and the success rate is 100%, proving that ACED can be applied to off-policy RL algorithms and achieve higher sample efficiency.

The horizontal axis in Figure 2 represents the number of demonstration trajectories $|\mathcal{T}|$ and the total number of curricula C_{max}, and the vertical axis represents the number of environment steps the training takes to converge averaged from 10 runs with different random seeds. From Figure 2 we can see that for ACED with BC, $|\mathcal{T}|$ has a significant impact on the number of environment steps it takes to converge. One potential reason that can cause this is, with a more diverse set of initializations during the curriculum training phase, ACED will better generalize to unseen random initial states and goal states after switching to normal rollout worker. Another potential reason is that BC is usually less prone to overfitting when the number of demonstrations is large, hence it should be able to generate better initial policies during pre-training with a larger $|\mathcal{T}|$. In comparison, the number of total curricula has less impact on the convergence performance of ACED with BC, but choosing a reasonable C_{max} can help accelerate training especially when the number of demonstrations is small. For the pick-and-place task we tested on, $C_{max} = 5$ generally performs better across different $|\mathcal{T}|$ values in terms of both the mean and the 90% confidence interval.

If we compare the convergence performance of ACED with BC and ACED without BC in Figure 2, we can see that ACED without BC generally takes longer to converge except for when there is only 1 demonstration trajectory. This shows that BC pre-training can provide a good initial policy and accelerate ACED training when sufficient demonstration trajectories are provided. However, when $|\mathcal{T}|$ is too small, BC pre-training might adversely affect ACED's performance. Another observation from Figure 2 is that the convergence performance of ACED without BC does not show a clear trend as $|\mathcal{T}|$ decreases, which means that the increasing trend we see in ACED with BC experiments is more likely to have been caused more by BC rather than ACED itself. One explanation for this observation is that, despite ACED's better generalization performance when $|\mathcal{T}|$ is large, the curriculum training phase itself can become more challenging and takes longer to converge with a larger $|\mathcal{T}|$. This

Table 1: Pick-and-Place Success Rate with PPO

Algorithm	Number of Curricula[1]	Number of Demonstrations				
		$\lvert\mathcal{T}\rvert = 100$	$\lvert\mathcal{T}\rvert = 50$	$\lvert\mathcal{T}\rvert = 20$	$\lvert\mathcal{T}\rvert = 5$	$\lvert\mathcal{T}\rvert = 1$
ACED with BC	$C_{max} = 8$	99%	100%	99%	97%	96%
	$C_{max} = 5$	96%	99%	99%	100%	95%
	$C_{max} = 3$	100%	99%	100%	100%	99%
	$C_{max} = 1$	100%	99%	100%	98%	99%
	Average[2]	98.8%	99.3%	99.5%	98.8%	97.3%
ACED without BC	$C_{max} = 5$	100%	95%	100%	93%	97%
BC Policy[3]		60%	54%	24%	2%	4%
Expert Demonstrations		92%	98%	95%	80%	100%

[1] For each set of experiment, we have 10 runs with different random seeds. For each run, we rollout 10 trajectories with the policy at convergence and compute the success rate, hence each entry is computed from a total of 100 rollout trajectories.
[2] The average success rate for $C_{max} = 8$, $C_{max} = 5$, $C_{max} = 3$ and $C_{max} = 1$.
[3] The success rate of the initial policy pre-trained by BC evaluated on 100 rollout trajectories.

is because ACED faces a more diverse set of initializations when there are more demonstration trajectories. Our experiments show that without BC pre-training, ACED actually performs the best with only 1 demonstration in pick-and-place tasks.

Table 1 compares the success rate of ACED during test time with the success rate of expert demonstrations and behavior cloning. We can see that even though the expert demonstrations aren't perfect (i.e. mostly have a success rate of less than 100%), ACED is able to learn the pick-and-place task with better-than-expert performance. This is because our approach doesn't rely on the expert policy except for the BC pre-training, and it instead tries to come up with its own policy that reaches the goal from states along demonstrations. Therefore, even with suboptimal demonstrations, ACED can still achieve better-than-expert performance. On the other hand, with policies trained only by BC, the success rate is much lower especially when the number of demonstrations is small, proving that our approach utilizes expert demonstrations in a more effective way than BC does. Another observation from Table 1 is that ACED generally achieves higher success rate with more demonstrations, but it is notable that even with 1 demonstration trajectory, it can still achieve a success rate of 96%. This is very encouraging because unlike many other imitation learning approaches that require a large number of demonstrations to work effectively, ACED can succeed with as few as 1 demonstration.

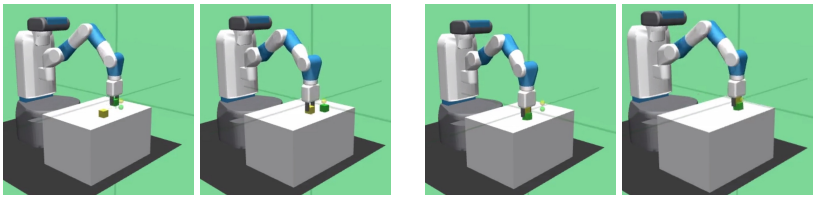
5.2 Block Stacking Tasks

ACED with BC is also evaluated in block stacking tasks with $|\mathcal{T}| = 100, 20$ and $C_{max} = 12, 8$, and its performance is compared with other state-of-the-art automatic curriculum methods including reverse curriculum [16] and the Montezuma's Revenge method [12]. Unfortunately, neither of the baseline approaches are able to successfully learn the stacking task (i.e. converge), hence we cannot compare with their convergence performance. Table 2 presents the average number of environment steps ACED with BC takes to converge in each set of experiments, and compares its success rate with the initial policies pre-trained by BC and with the expert demonstrations. Interestingly, we observe that the number of environment steps until convergence is much lower when $|\mathcal{T}| = 20$ compared to when $|\mathcal{T}| = 100$. We believe this is because ACED with BC has converged to two different policies for the two sets of experiments with different $|\mathcal{T}|$ values. From the recorded videos we found that in all 10 runs with $|\mathcal{T}| = 100$ (including $C_{max} = 12$ and $C_{max} = 8$), the policies ACED with BC converged to are similar to the demonstrations, where the robot first picks up the green block and places it onto the goal, and then picks up the yellow block and places it onto the green block. However, in all 10 runs with $|\mathcal{T}| = 20$, the policy ACED with BC converged to takes a different route: it first places the yellow block on top of the green block, and then picks up the two blocks together to place them onto the goal. Figure 3 illustrates the two different policies visually by showing two representative frames from each rollout video. This finding shows that, with a smaller number of demonstrations, ACED with BC has more flexibility to come up with novel solutions instead of following the demonstration trajectories. Because the $|\mathcal{T}| = 20$ solution takes fewer steps, it is easier

Table 2: Block Stacking Performance with PPO

Number of Demonstrations		$\lvert\mathcal{T}\rvert = 100$		$\lvert\mathcal{T}\rvert = 20$	
ACED with BC[1]	Total Curriculum Number	$C_{max} = 12$	$C_{max} = 8$	$C_{max} = 12$	$C_{max} = 8$
	Convergence Env Steps[2]	213.08	169.88	143.02	119.79
	Success Rate[3]	100%	100%	96%	100%
BC Policy Success Rate[4]		0%		0%	
Expert Demonstration Success Rate[5]		84%		85%	

[1] Each set of experiment for ACED with BC is averaged from 5 runs with different random seeds.
[2] Presented in millions. The training for stacking tasks is more unstable, so we separated the training process into two sections: 1) use curriculum rollout workers to train until all parallel workers reach $C = C_{max}$ and 2) set all parallel workers to be normal rollout workers and train until convergence. The convergence environment steps presented here are the sum of the two sections.
[3] The success rate for ACED with BC is evaluated with 10 rollout trajectories per random seed (50 total).
[4] The success rate of the initial policy pre-trained by BC evaluated on 100 rollout trajectories.
[5] The demonstrations for stacking have much lower success rates because there are two blocks in the scene and one block might obstruct the straight line policy the moves the other block to its goal pose. Since the straight line policies are open-loop, the agent can't recover from such failures.



(a) Solution policy when $\lvert\mathcal{T}\rvert = 100$ (b) Solution policy when $\lvert\mathcal{T}\rvert = 20$

Figure 3: Two different stacking policies ACED with BC converged to with different $\lvert\mathcal{T}\rvert$.

to train and is more robust when generalizing to new initial and goal poses. We also observed that the $\lvert\mathcal{T}\rvert = 20$ training curves experience less performance drop when switching from curriculum rollout workers to normal rollout workers. We believe this is why ACED with BC converges faster when trained with 20 demonstrations.

Another finding from Table 2 is that for the block stacking task, $C_{max} = 8$ has better performance than $C_{max} = 12$ in terms of both convergence speed and success rate. Compared to the number of demonstrations, the number of total curricula has less impact on ACED with BC's solutions and training performance, and different C_{max} didn't cause the solution policies to differ qualitatively.

We show the comparison of ACED with two state-of-the-art automatic curriculum generation methods [16; 12] in the block stacking task in Figure 4. We refer to the approach presented in [16] the reverse curriculum method, and the one presented in [12] the Montezuma's Revenge method. In Figure 4, ACED is implemented with PPO and is provided with 20 demonstrations and BC pre-training. As we mentioned earlier, neither of the two baseline automatic curriculum methods are able to converge in the block stacking tasks. In all runs of the reverse curriculum method, the start state distribution has not moved to the true S_0 after 400 million environment steps of training, and achieves 0% success rate during testing. In all runs of the Montezuma's Revenge method, moving through the curriculum rollout workers are relatively quick, but the learning curve drops to zero and never goes back up once it switches to the normal rollout worker. This shows that in challenging tasks in continuous state space, policies training using a fixed demonstration without randomization struggle to generalize to the entire initial state distribution S_0 and goal distribution S_g. Figure 4 shows the learning curves of one representative run for each of the curriculum generation methods.

6 Discussion

This paper presents Automatic Curricula via Expert Demonstrations, an RL approach that combines ideas from both imitation learning and curriculum learning to tackle challenging robotics manipulation tasks with sparse reward signals. Through resetting the training episodes to states along demonstration

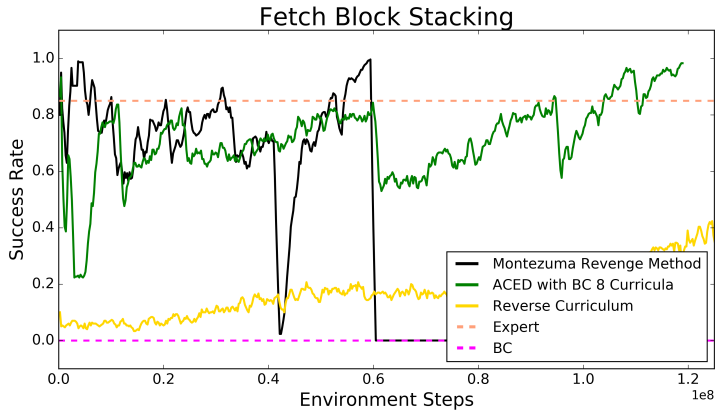


Figure 4: Learning curves of ACED with BC, the reverse curriculum method [16], and the Montezuma's Revenge method [12] in the block stacking task.

trajectories, ACED is able to control the difficulty of the tasks by moving the reset states from the end of the demonstration to the beginning based on the learning progress of the RL agent. This procedure naturally forms a curriculum and makes challenging exploration problems feasible to learn. One main advantage of ACED is that it only requires demonstration states and not actions when deployed on its own. ACED can also be intuitively combined with many existing imitation learning approaches to utilize expert demonstrations more efficiently, including adding demonstrations to replay buffers [32], introducing GAN-based rewards[10], and using behavior cloning to pre-train the policies. In this paper, a version of ACED with policies pre-trained via behavior cloning is compared with ACED on its own as an example. We evaluate the performance of ACED on block pick-and-place tasks and stacking tasks, and show that pick-and-place can be learned with as few as 1 demonstration and stacking can be learned with 20 demonstrations. We also analyzed the impact of the number of demonstration trajectories and the total number of curricula on ACED's performance, and discovered that ACED can learn novel solutions that are very distinct from expert demonstrations when the number of demonstrations is small.

References

[1] L. P. Kaelbling and T. Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013.

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[3] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, 2017.

[4] C. Colas, P.-Y. Oudeyer, O. Sigaud, P. Fournier, and M. Chetouani. Curious: Intrinsically motivated modular multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 1331–1340, 2019.

[5] S. Dai, W. Xu, A. Hofmann, and B. C. Williams. An Empowerment-based Solution to Robotic Manipulation Tasks with Sparse Rewards. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.001.

[6] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[7] T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 1851–1860. PMLR, 2018.

[8] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[9] B. Ivanovic, J. Harrison, A. Sharma, M. Chen, and M. Pavone. Barc: Backward reachability curriculum for robotic reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 15–21. IEEE, 2019.

[10] J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:4565–4573, 2016.

[11] X. B. Peng, A. Kanazawa, S. Toyer, P. Abbeel, and S. Levine. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. In *International Conference on Learning Representations*, 2018.

[12] T. Salimans and R. Chen. Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018.

[13] C. Resnick, R. Raileanu, S. Kapoor, A. Peysakhovich, K. Cho, and J. Bruna. Backplay:" man muss immer umkehren". *arXiv preprint arXiv:1807.06919*, 2018.

[14] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1311–1320. JMLR. org, 2017.

[15] P. Mirowski, M. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, A. Zisserman, R. Hadsell, et al. Learning to navigate in cities without a map. In *Advances in Neural Information Processing Systems*, pages 2419–2430, 2018.

[16] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning*, pages 482–495, 2017.

[17] R. Wang, J. Lehman, J. Clune, and K. O. Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

[18] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528, 2018.

[19] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=SkT5Yg-RZ`.

[20] D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

[21] M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[22] M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019. doi: 10.15607/RSS.2019.XV.031.

[23] D. Wang, C. Devin, Q.-Z. Cai, F. Yu, and T. Darrell. Deep object-centric policies for autonomous driving. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8853–8859. IEEE, 2019.

[24] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018.

[25] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[26] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[27] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58, 2016.

[28] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[29] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[30] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[31] B. Kang, Z. Jie, and J. Feng. Policy optimization with demonstrations. In *International Conference on Machine Learning*, pages 2469–2478. PMLR, 2018.

[32] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[33] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[34] B. Eysenbach, S. Gu, J. Ibarz, and S. Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=S1vuO-bCW`.

[35] K. Xu, S. Verma, C. Finn, and S. Levine. Continual learning of control primitives : Skill discovery via reset-games. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4999–5010, 2020.

[36] M. Turchetta, A. Kolobov, S. Shah, A. Krause, and A. Agarwal. Safe reinforcement learning via curriculum induction. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12151–12162, 2020.

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

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

[39] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[40] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[41] J. B. Lanier, S. McAleer, and P. Baldi. Curiosity-driven multi-criteria hindsight experience replay. *arXiv preprint arXiv:1906.03710*, 2019.

[42] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.