DIVIDE AND CONQUER REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Standard model-free deep reinforcement learning (RL) algorithms sample a new initial state for each trial, allowing them to optimize policies that can perform well even in highly stochastic environments. However, problems that exhibit considerable initial state variation typically produce high-variance gradient estimates for model-free RL, making direct policy or value function optimization challenging. In this paper, we develop a novel algorithm that instead optimizes an ensemble of policies, each on a different "slice" of the state space, and gradually unifies them into a single policy that can succeed on the whole state space. This approach, which we term divide and conquer RL, is able to solve complex tasks where conventional deep reinforcement learning methods are ineffective. Our results show that divide and conquer RL greatly outperforms conventional policy gradient methods on challenging grasping, manipulation, and locomotion tasks, and exceeds the performance of a variety of prior methods.

1 INTRODUCTION

Deep reinforcement learning (RL) algorithms have demonstrated an impressive potential for tackling a wide range of complex tasks, from game playing (Mnih et al., 2015) to robotic manipulation (Levine et al., 2016; Kumar et al., 2016; Popov et al., 2017; Andrychowicz et al., 2017). However, many of the standard benchmark tasks in reinforcement learning, including the Atari benchmark suite (Mnih et al., 2013) and all of the OpenAI gym continuous control benchmarks (Brockman et al., 2016) lack the kind of diversity that is present in realistic environments. Many of the most challenging real-world domains that might benefit from deep reinforcement learning are characterized by diversity and variability, and one of the most compelling use cases for such algorithms is to create autonomous agents that can interact intelligently with varied and stochastic environments. However, such environments present a major challenge for current algorithms. The most standard avenue for incorporating variability into RL training is to choose a stochastic initial state distribution for the underlying Markov decision process. Highly stochastic initial state distributions lead to high-variance policy gradient estimates, which in turn hamper effective learning. Similarly, variability can also be incorporated via picking distribution over the goal state.

In this paper, we explore RL algorithms that are especially well suited for tasks with a high degree of variability in the state space, both initial and goal state . We argue that a large class of practically interesting real-world problems fall into this category, but current RL algorithms are poorly equipped to handle them, as illustrated in our experimental evaluation. Our main observation is that, for tasks with a high degree of state variability, it is often much easier to obtain effective solutions to individual parts of the state space and then merge these solutions into a single policy, than to solve the entire task as a monolithic stochastic MDP. To that end, we can partition the initial state distribution into a set of distinct "slices," and train a separate policy for each slice. For example, if we imagine the task of picking up a block with a robotic arm, different slices might correspond to different initial positions of the block. For each slice, the algorithm might train a different policy with a distinct strategy. By employing a combination of mutual KL-divergence constraints and supervised distillation, we can gradually merge these distinct policies into a single global policy, which succeeds on all of the initial states at convergence.

It may at first seem surprising that this procedure provides benefit. After all, if the final global policy can solve the entire task, then surely each local policy also has the representational capacity to capture a strategy that is effective on the entire state space. However, it is worth considering that a policy in a reinforcement learning algorithm must be able to represent not only the final optimal

policy, but also all of the intermediate policies during learning. By decomposing these intermediate policies over the different slices of the state space, our method enables effective learning even on tasks that with very diverse initial state distributions. This strategy also benefit from the fact that the gradient information can better estimated in the local slices with less number of samples leading to accelerated learning. Intermediate supervised distillation step helps share information between the local policies. Information sharing accelerates learning for slow learning policies, and help policies avoid local minimas.

The main contribution of our paper is a reinforcement learning algorithm for tasks with diverse initial states, which we term divide and conquer (DnC) reinforcement learning. We present our algorithm and motivate its mathematical formulation, and show a detailed comparative evaluation of this approach against standard reinforcement learning methods and prior techniques. We demonstrate a substantial improvement in performance on torque-controlled robotic grasping and manipulation, and a variety of difficult locomotion scenarios.

2 Related Work

Prior work has addressed reinforcement learning with diverse behaviors, both in locomotion (Heess et al., 2017) and manipulation (Rajeswaran & V. Kumar, 2017; Andrychowicz et al., 2017; Nair et al., 2017). However, these methods typically make a number of simplifications, such as the use of demonstrations to help guide reinforcement learning (Rajeswaran & V. Kumar, 2017; Nair et al., 2017), or the use of a higher-level action representation, such as Cartesian end-effector control for a robotic arm (Andrychowicz et al., 2017; Nair et al., 2017). Here we demonstrate that the proposed algorithm can learn complex grasping task that requires picking up an object from various positions. We demonstrate the strength of DnC strategy by learning behaviours directly in the low level torque action space, without the need of demonstration or high level action representation. The tasks selection in this work is significantly more complex than the taks leveraging cartesian position action spaces in prior work (Andrychowicz et al., 2017; Nair et al., 2017) or the relatively simple picking setup proposed by Popov et al. (2017), which consists of a single position and a variety of shaping rewards. In the domain of locomotion, we show that our approach substantially outperforms the direct policy search method proposed by Heess et al. (2017).

Our method is related to guided policy search (GPS) algorithms (Levine & Koltun, 2013; Levine et al., 2016). These algorithms train several "local" policies by using a trajectory-centric reinforcement learning method, and a single "global" policy, typically represented by a deep neural network, which attempts to mimic the local policies. The local policies are constrained against the global policy, typically via a KL-divergence constraint. Our method also trained local policies, though the local policies are themselves represented by , more flexible, nonlinear neural network policies. Use of neural network policies significantly improves representation power of the individual controllers and facilitates the use of various off the shelf reinforcement learning algorithms. Furthermore, we constrain the various policies against one another, rather than against a single central policy, which we find substantially improves performance, as discussed in Section 5.

Along similar lines, Teh et al. (2017) propose an approach similar to GPS for the purpose of transfer learning, where a single policy is trained to mimic the behavior of policies trained in different domains. Our approach resembles GPS, in that we decompose a single complex task into local pieces, but also resembles Teh et al. (2017), in that we use nonlinear neural network local policies. Although Teh et al. (2017) propose a method intended for transfer learning, we adapted it to our problem formulation for comparison. We present results demonstrating that our approach substantially outperforms the method proposed by Teh et al. (2017).

3 Preliminaries

An episodic Markov Decision Process (MDP) is classically modelled as $\mathbf{M} = (\mathcal{S}, \mathcal{A}, P, r, \rho)$ where $\mathcal{S}, \mathcal{A}$ are continuous sets of states and actions respectively. $P(s', s, a)$ is the transition probability distribution,$r : \mathcal{S} \rightarrow \mathbb{R}$ the reward function, and $\rho : S \rightarrow \mathbb{R}_+$ an initial state distribution. We consider an alternative formulation of an MDP, where the initial state distribution is conditioned on some variable ω, which we refer to as a "context." Formally, $\Omega = (\omega_i)_{1...n}$ is a finite set of contexts, and $\rho : \Omega \times S \rightarrow \mathbb{R}_+$ a joint distribution over contexts ω and initial states s_0. One can imagine

that sampling initial states is a two stage process: first, contexts are sampled as $\rho(\omega)$, and then initial states are drawn given the sampled context as $\rho(s|\omega)$. Note that for an arbitrary MDP, one can embed the context into the state as an additional state variable with indipendent distribution structure so that $P(s_0, \omega) = P(s_0|\omega)P(\omega)$. A trajectory in this MDP can be written as $\tau = (\omega, s_0, a_0, s_1, a_1, \dots)$. We hope to find a stochastic policy $\pi : \mathcal{S}, \mathcal{A} \to \mathbb{R}_+$ which maximizes the expected reward of the policy: $\eta(\pi) = \mathbb{E}_{\tau\sim\pi}[r(\tau)]$, where $r(\tau) = \sum_{i=0}^{n} r(s_i)$.

4 Divide and Conquer Reinforcement Learning

In this section, we derive our divide and conquer reinforcement learning algorithm. We first motivate the approach by describing a policy learning framework for the contextual MDP setting described above, and then introduce a practical algorithm that can implement this framework for complex deep reinforcement learning problems.

4.1 Learning Policies in Contextual MDPs

The addition of contextual structure allows for two immediate extensions of our given MDP $\mathbf{M}$. First, we consider an augmented MDP $\mathbf{M}'$ where each state contains information about the context $(\mathcal{S} \times \Omega, \mathcal{A}, P, r, \rho)$, but neither the dynamics nor rewards take context into account. A trajectory in this MDP is $\tau = ((\omega, s_0), a_0, (\omega, s_1), a_1, \dots)$. The second class of MDPs are the context restricted MDPs: for a context ω, we have $\mathbf{M}_\omega = (\mathcal{S}, \mathcal{A}, P, r, \rho_\omega)$, where $\rho_\omega(s) = \rho(s|\omega)$, i.e. the context is fixed to ω for each trajectory.

A stochastic policy π in the augmented MDP $\mathbf{M}'$ can be decoupled into a family of simpler stochastic policies $\pi = (\pi_i)_{1\dots n}$, where $\pi_i : \mathcal{S}, \mathcal{A} \to [0, 1]$, and $\pi((\omega_i, s), a) = \pi_i(s, a)$. Notice that π_i is a policy for the context-restricted MDP $\mathbf{M}_{\omega_i}$, and that a family of optimal policies in the class of context-restricted MDPs induces an optimal policy π in $\mathbf{M}'$, and vice versa.This implies that policy search in the augmented MDP can be decomposed into policy search in the context-restricted MDPs.

Given a stochastic policy in the augmented MDP $(\pi_i)_{1\dots n}$, we can induce a stochastic policy π_c in the original MDP, by defining $\pi_c(s, a) = \sum_{\omega\in\Omega} p(\omega|s)\pi_\omega(s, a)$, where $p(\cdot|s)$ is a belief distribution of what context the trajectory is in. Our insight is that it is important for each local policy to not only be good for the designated instance, but also be capable of working in other instances. One reason is that it might be hard to accurately infer the context at deployment time on the original MDP, and local policies that work more broadly can compensate for errors in inference. Secondly, requiring that local policies be capable of working broadly for multiple contexts allows for sharing of information so that local policies designated for hard contexts can bootstrap their solutions off easier contexts. In the extreme case, where local policies work very broadly, we obtain policies that are capable of operating even if no context information is provided. From here on, we refer to π_c as the central or global policy, and π_i as the context-specific and local policies.

In order to find the optimal policy for the original MDP, we search for a policy $\pi = (\pi_i)_{1\dots n}$ in the augmented MDP that maximizes $\eta(\pi) - \alpha\mathbb{E}_\pi[D_{KL}(\pi\|\pi_c)]$: maximizing expected reward for each instance while remaining close to a central policy. This allows the central policy π_c to work for all the contexts, thereby transfering to the original MDP. Using Jensen's inequality to bound the KL divergence between π and π_c, we minimize the right hand side as a bound for minimizing the intractable KL divergence optimization problem.

$$\mathbb{E}_\pi[D_{KL}(\pi\|\pi_c)] \leq \sum_{i,j} \rho(\omega_i)\rho(\omega_j)\mathbb{E}_{\pi_i}[D_{KL}(\pi_i\|\pi_j)] \tag{1}$$

4.2 The Divide and Conquer Reinforcement Learning Algorithm

We now present a policy optimization algorithm which takes advantage of this contextual starting state, in a manner illustrated by the theory. In particular, our approach to finding a good policy π_c for a task involves finding local policies $(\pi_i)_{1\dots n}$ that maximize expected reward on the individual contexts, while remaining constrained to be similar to one another. We modify policy gradient algorithms for optimizing local policies with these requirements. Policy gradient methods directly optimize the parameters of a stochastic policy through local gradient based methods. Taking gradient steps using the score function gradient estimator yields the REINFORCE algorithm

(Williams, 1992), whereas taking the curvature of the parameter space into account is the basis for NPG (Kakade, 2002). We base our algorithm on TRPO (Schulman et al., 2015), a policy gradient method which takes gradient steps according to a surrogate loss

$$\mathcal{L}(\pi) = -\mathbb{E}_{\pi}\left[A(s,a)\frac{\pi(a|s)}{\pi_{old}(a|s)}\right] \tag{2}$$

while constraining the mean divergence from the old policy to some step size. We choose TRPO for it's strong convergence guarantees and excellent practical performance on high-dimensional continuous control problems.

In our framework, we are attempting to optimize $\eta(\pi) - \alpha\mathbb{E}_{\pi}[D_{KL}(\pi\|\pi_c)]$, where α determines the relative balancing effect of expected reward and deviation, which means a surrogate loss of form

$$\mathcal{L}(\pi_1 \dots \pi_n) = -\sum_{i=1}^{n}\mathbb{E}_{\pi_{i,old}}\left[A(s,a)\frac{\pi_i(a|s)}{\pi_{i,old}(a|s)}\right] + \alpha\left(\sum_{i,j}\rho(\omega_i)\rho(\omega_j)\mathbb{E}_{\pi_i}\left[D_{KL}(\pi_i\|\pi_j)\right]\right) \tag{3}$$

There are some interesting nuances in this formulation; while each local policy π_i' attempts to maximize its expected reward on the particular context, the KL divergence terms constrain the policy to not deviate too far from the other π_j' on it's own context, while also encouraging the policy to be more similar to the other local policies on their respective tasks. Looking at the terms for which a specific π_i corresponds to, this decomposition becomes evident.

$$\mathcal{L}(\pi) \propto_{\pi_i} \underbrace{-\mathbb{E}_{\pi_{i,old}}\left[A(s,a)\frac{\pi_i(a|s)}{\pi_{i,old}(a|s)}\right]}_{\text{Maximizes } \eta(\pi_i)} + \alpha\rho(\omega_i)\sum_j \rho(\omega_j)\big(\underbrace{\mathbb{E}_{\pi_i}[D_{KL}(\pi_i\|\pi_j)]}_{\text{Constraint on own context}} + \underbrace{\mathbb{E}_{\pi_j}[D_{KL}(\pi_j\|\pi_i)]}_{\text{Constraint on other contexts}}\big) \tag{4}$$

On each iteration, the algorithm collects trajectories from each context-restricted MDP $\mathbf{M}_{\omega_i}$ using π_i, and each local policy π_i does a TRPO update with the surrogate loss in an alternating schema (i.e. in succession). Unlike doing a joint TRPO update on surrogate loss over all the local policies, this alternating scheme ensures that no context dominates one another in the update process. This process is iterated a large number of times, ultimately resulting in local policies $(\pi_i)_{1\dots n}$. We must now retrieve π_c, a central policy in the original context from the local policies, which is done by minimizing the KL divergence between π and π_c, which neatly simplifies into a maximum likelihood problem with demonstrations from all the various policies.

$$\mathcal{L}_{center}(\pi_c) = \mathbb{E}_{\pi}[D_{KL}(\pi(\cdot|s)\|\pi_c(\cdot|s))] \propto \sum_i \rho(\omega_i)\mathbb{E}_{\pi_i}\left[-\log\pi_c(s,a)\right] \tag{5}$$

If the π_c doesn't solve the full problem, we iterate this procedure by running through the training of the $\pi_1 \dots \pi_n$ again, this time however initializing the π_i's to start at π_c, until the training of the subtasks completes.We continually repeat this alternating local policy optimization and global policy optimization until convergence.

```
function DNC(Contexts ω_1 ... ω_n)
    T ← Number of Training Loops
    R ← Distillation Frequency
    Randomly initialize central policy π_c
    for T iterations do
        Set π_i = π_c for all i = 1 ... n
        for j ← 1, ... R do
            Collect trajectories T_i on context ω_i using policy π'_i for all i = 1 ... n
            for all local policies π_i do
                Take TRPO update step minimizing surrogate loss function L wrt π_i
            end for
        end for
        Minimize L_center w.r.t. π_c (using previously sampled states)
    end for
    return π_c
end function
```

5 EXPERIMENTAL EVALUATION

We focus our analysis on tasks spanning two different domains – manipulation and locomotion. Manipulation tasks involve handling a free floating object with a robotic arm and locomotion tasks involve tackling challenging terrains. Be illustrate a variety of behaviors in both settings. Tasks were designed to exhibit complex contact rich behaviors in settings with considerable state variation. All of our environments are designed and simulated in Mujoco (Todorov et al., 2012).

Our experiments and analysis aim to address the following questions:

1. Can DnC solve highly complex tasks in a variety of domains, especially tasks that cannot be solved with current conventional policy gradient methods?
2. How does the form of the constraint on the ensemble policies in DnC affect the performance of the final policy, as compared to previously proposed constraints?

We compare **DnC** to the following prior methods and ablated variants:

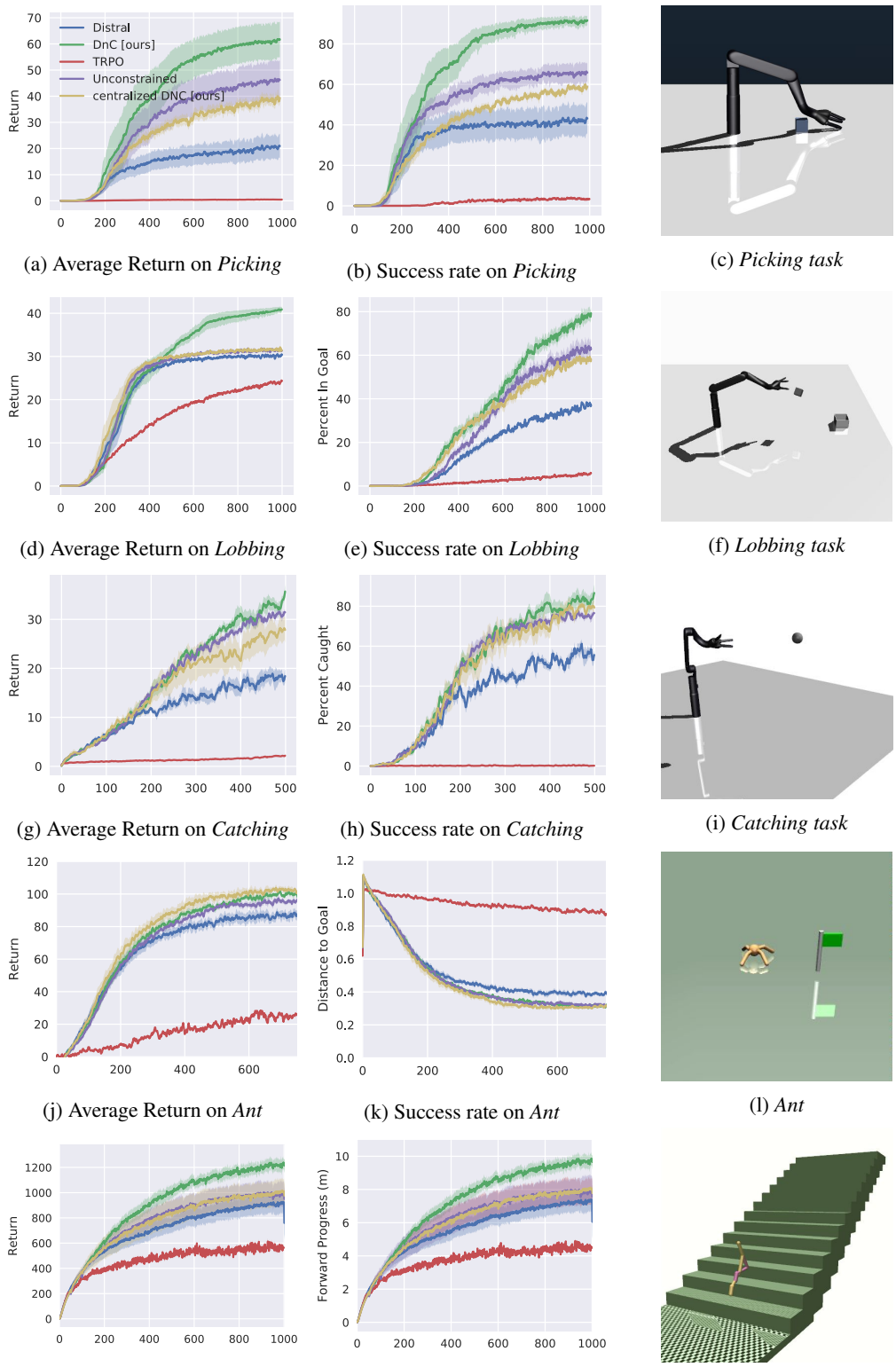
- **TRPO.** TRPO (Schulman et al., 2015) represents a state-of-the-art policy gradient method, which we use for the standard RL comparison without decomposition into contexts. TRPO is provided with the same batch size as the sum of the batches over all of the policies in our algorithm, to ensure a fair comparison.
- **Distral.** Originally formulated as transfer learning in a discrete action space (Teh et al., 2017), we extend Distral to a continuous setting, where each context ω as a different task. This algorithm, which resembles the structure of guided policy search, also trains an ensemble of policies, but constrains them at each gradient step against a single global policy trained with supervised learning, and omits the distillation step that our method performs every R iterations (we use $R = 100$ in our experiments).
- **Unconstrained DnC.** We run the DnC algorithm without any KL constraint. This reduces to running TRPO to train policies $(\pi_i)_{1\ldots n}$ on each context, and distilling the resulting local policies every R iterations.
- **Centralized DnC.** Whereas DnC derives its KL constraint from the assumption that the policy need not distinguish between contexts, centralized DnC uses the reverse motivation, where the algorithm perfectly identifies ω from s by an oracle. Deriving the theory from this formulation, the algorithm is equivalent to the Distral objective, however distills every R steps.

6 ROBOTIC MANIPULATION

For robotic manipulation, we simulate the Kinova Jaco, a 7 DoF robotic arm with 3 fingers. In all of the following tasks, the agent receives full state information, which includes the current absolute location of external objects like boxes. The agent uses low level joint torque control to perform the required actions. Note that use of low level torque control significantly increases the complexity of the task, as raw torque control on a 7 DoF arm requires delicate movements of the joints to perform each task.

Picking. The *Picking* task requires the Jaco to pick up a small block and raise it as high as possible. The reward signal is sparse: the agent receives reward only when the block is in the agent's hand. The observation space has 34 dimensions, and the action space has 7. The starting position of the block can be at any point within a fixed 30cm x 30cm square, so we attempt to find a policy which can pick up from anywhere in this region. Picking up the block from different locations within the workspace require diverse poses, making this task challenging in the torque control framework. TRPO can only solve the picking task from a 4cm x 4cm workspace. In larger workspaces, the Jaco fails to grasp from the edges of the workspace.

Lobbing. The *Lobbing* task requires the Jaco to flick a block into a small box, which is placed far enough that the arm cannot reach it directly. Rewards are awarded at the end of a trajectory if the block left the ground (i.e., wasn't pushed) based on the distance from the location of the initial landing to the box. The location of the target box is randomized over a 1m x 1m square. The observation space has 40 dimensions, and the action space has 7. This problem inherits many challenges



(a) Average Return on *Picking*

(b) Success rate on *Picking*

(c) *Picking task*

(d) Average Return on *Lobbing*

(e) Success rate on *Lobbing*

(f) *Lobbing task*

(g) Average Return on *Catching*

(h) Success rate on *Catching*

(i) *Catching task*

(j) Average Return on *Ant*

(k) Success rate on *Ant*

(l) *Ant*

(m) Average Return on *Stairs*

(n) Forward Progress on *Stairs*

(o) *Stairs*

Figure 1: Average returns and success rate on Picking, Lobbing, Catching, Ant, and Stairs. On all of the tasks, DnC RL achieves the best results. On the Ant task, the centralized variant achieves slightly better final performance, while on the Picking, Lobbing, and Stairs task the full algorithm outperforms all others by a wide margin.

	Picker	Lobber	Catcher	Ant Position	Stairs
TRPO	0.3 ± 0.2	22 ± 1.2	8.2 ± 2.1	20.1 ± 3.2	503.3 ± 41.2
Distral	22.0 ± 4.5	30.4 ± 0.5	17.9 ± 1.7	87.9 ± 3.2	858.5 ± 39.3
Unconstrained	44.3 ± 8.7	31.4 ± 0.4	32.3 ± 0.9	92.6 ± 2.4	971.1 ± 27.5
Centralized DnC (ours)	37.3 ± 2.5	31.9 ± 0.4	27.1 ± 2.5	$\mathbf{102.6 \pm 3.9}$	951.5 ± 24.2
DnC (ours)	$\mathbf{61.9 \pm 9.2}$	$\mathbf{38.6 \pm 1.7}$	$\mathbf{37.1 \pm 1.1}$	98.0 ± 3.1	$\mathbf{1157.4 \pm 26.3}$

Table 1: Overall performance comparison between DnC and competing methods, based on final average return. Performance varies from run to run, so we run each experiment with 3 random seeds. For each of the tasks, the best performing method is DnC or centralized DnC.

from the picking task. Furthermore, the sequential nature of grasping and flicking necessitates that information pass temporally and requires synthesis of multiple skills.

Catching. In the *Catching* task, a ball is thrown at the robot, and the arm must catch it in the air. Fixed reward is awarded every step that the ball is in or next to the hand. The initial position and velocity of the ball are appropriately randomized for each trajectory.

DnC exceeds the performance of the alternative methods on all of the manipulation tasks, as shown in Figure 1. For each task, we include the average reward, as well as a success rate measure, which provides a more interpretable impression of the performance of the final policy. TRPO by itself is unable to solve any of the tasks, with success rates below 10% in each case. The policies learned by TRPO are qualtitatively reasonable, but lack the intricate control required for success.

TRPO fails because of the high stochasticity in the problem and the diversity of optimal behaviour for various initial states, because the algorithm cannot make progress on the full task with such noisy gradients. When we partition the manipulation tasks into contexts, the behavior within each context is much more homogeneous. All three tasks are partitioned into four contexts along grids for the initial state of the object (block or target bin for lobbing). Figure 1 shows that, when trained with this choice of context, DnC outperforms both the adapted Distral variant and the two ablations of our method. On the picking task, DnC has a 16% higher success rate than the next best method, which is an ablated variant of DnC, and on the lobbing task it is better by 10%. Both the pairwise KL penalty and the periodic reset in DnC appear to be crucial for the algorithm's performance. In contrast to the methods that share information exclusively though a single global policy, the pairwise KL terms allow for more efficient information exchange. On the Picking task, the centralized variant of DnC struggles to pick up the object pockets along the boundaries of the contexts, likely because the local policies differ too much in these regions, and centralized distillation is insufficient to produce effective behavior.

On the catching task (Figure 3c), the baselines which distill every 100 iterations (the DnC variants) all perform well, whereas Distral lags behind. The policy learned by Distral grasps the ball from an awkward orientation, so the grip is unstable and the ball quickly drops out. Since Distral doesn't distill and reset the local policies, it fails to escape this local optimal behaviour.

7 Locomotion

Our locomotion tasks involve learning parameterized navigation skills in two domains.

Ant Position In the *Ant Position* task, the quadruped ant is tasked with reaching a particular goal position; the exact goal position is randomly selected along the perimeter of a circle 5m in radius for each trajectory. The Ant is penalized for its distance from the goal every timestep. Although moving the ant in a single direction is solved, training an ant to walk to an arbitrary point is difficult because the task is symmetric and the global gradients may cancel one another out, especially in the presence of such a

Stairs In the *Stairs* task, a planar (2D) bipedal robot must climb a set of stairs, where the stairs have varying heights and lengths. The agent is rewarded for forward progress. Unlike the other tasks in this paper, there exists a single gait that can solve all possible heights, since a policy that can clear the highest stair can also clear lower stairs with no issues. However, optimal behavior

that maximizes reward will maintain more specialized gaits for various heights. The agent locally observes the structure of the environment via a perception system that conveys the information about the height of the next step.

As in manipulation, we find that DnC performs either on par or better than all of the alternative methods on each task. TRPO is technically able to solve the ant position task with 5000 iterations, however at significant cost (roughly 400 million timesteps). Variants of our method solve the task in a tenth of the number of steps. Behaviour of TRPO within the timescale of the experiment has the ant moving in random directions throughout a trajectory, unable to clearly associate movement in a direction with the goal reward. On the *Stairs* task, TRPO learns to take long striding gaits that perform well on shorter stairs but cause the agent to trip on the taller stairs, because the reward signal from the shorter stairs is much stronger. In DnC, by separating the gradient updates by context, we can mitigate the effect of a strong reward signal on a context from affecting the policies of the other contexts.

Partitions and contexts for the ant task most naturally come by dividing the unit circle into segments; we split the initial state distribution into four quadrant contexts. The boundaries of these partitions are the ant's four legs, since gaits will be most similar when the same legs act as front legs. For the *Stairs* task, we choose to slice based on height, since the length of each stair doesn't affect the gait as much as height. In particular, we partition the heights into 5 contexts.

Although Figure (1n) seems to indicate that all the baselines learn similar policies at varying levels of efficiency, we notice large differences between the gaits learned by the baselines and DnC. DnC learns a striding gait on shorter stairs, and a jumping gait on taller stairs, but it is clearly visible that the two gaits share structure. In contrast, the other partitioning algorithms learn hopping motions that perform well on tall stairs, but are suboptimal on shorter stairs, so brittle to context.

8 Discussion and Future Work

In this paper, we proposed divide and conquer reinforcement learning, an RL algorithm that separates complex tasks into a set of initial conditions, each of which can be used to learn a separate policy. These separate policies are constrained against one another to arrive at a single, globally coherent solution, which can then be used to solve the task from any initial state. Our experimental results show that divide and conquer reinforcement learning substantially outperforms standard RL algorithms that sample initial states from the initial state distribution at each trial, as well as previously proposed methods that employ ensembles of policies. For each of the domains in our experimental evaluation, standard policy gradient methods are generally unable to find a successful solution.

Although our approach improves on the power of standard reinforcement learning methods, it does introduce additional complexity due to the need to train ensembles of policies. Sharing of information across the policies is accomplished by means of KL-divergence constraints, but no other explicit representation sharing is provided. A promising direction for future research to both reduce the computational burden and improve representation sharing is to train policies with both shared and separate components. Exploring this direction could yield methods that are more efficient both computationally and in terms of experience.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *CoRR*, abs/1707.01495, 2017.

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

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017.

Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pp. 1531–1538, 2002.

Vikash Kumar, Emanuel Todorov, and Sergey Levine. Optimal control with learned local models: Application to dexterous manipulation. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 378–383. IEEE, 2016.

Sergey Levine and Vladlen Koltun. Guided policy search. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pp. 1–9, 2013. URL `http://jmlr.org/proceedings/papers/v28/levine13.html`.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end learning of deep visuomotor policies. *Journal of Machine Learning Research (JMLR)*, 2016.

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.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. *CoRR*, abs/1709.10089, 2017.

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

A. Rajeswaran and J. Schulman E. Todorov S. Levine V. Kumar, A. Gupta. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *ArXiv e-prints*, 2017.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL `http://arxiv.org/abs/1502.05477`.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *CoRR*, abs/1707.04175, 2017. URL `http://arxiv.org/abs/1707.04175`.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pp. 5026–5033. IEEE, 2012. ISBN 978-1-4673-1737-5. URL `http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12`.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992696. URL `https://doi.org/10.1007/BF00992696`.