Deep Reinforcement Learning for Vision-Based Robotic Grasping: A Simulated Comparative Evaluation of Off-Policy Methods

Deirdre Quillen*[1], Eric Jang*[1], Ofir Nachum*[1], Chelsea Finn[2], Julian Ibarz[1], and Sergey Levine[1,2]

[1]Google Brain, [2]University of California, Berkeley

***Abstract*— In this paper, we explore deep reinforcement learning algorithms for vision-based robotic grasping. Model-free deep reinforcement learning (RL) has been successfully applied to a range of challenging environments, but the proliferation of algorithms makes it difficult to discern which particular approach would be best suited for a rich, diverse task like grasping. To answer this question, we propose a simulated benchmark for robotic grasping that emphasizes off-policy learning and generalization to unseen objects. Off-policy learning enables utilization of grasping data over a wide variety of objects, and diversity is important to enable the method to generalize to new objects that were not seen during training. We evaluate the benchmark tasks against a variety of Q-function estimation methods, a method previously proposed for robotic grasping with deep neural network models, and a novel approach based on a combination of Monte Carlo return estimation and an off-policy correction. Our results indicate that several simple methods provide a surprisingly strong competitor to popular algorithms such as double Q-learning, and our analysis of stability sheds light on the relative tradeoffs between the algorithms [1].**

I. Introduction

Robotic grasping is one of the most fundamental robotic manipulation tasks: before interacting with objects in the world, a robot typically must begin by grasping them. Prior work in robotic manipulation has sought to address the grasping problem through a wide range of methods, from analytic grasp metrics [43], [36] to learning-based approaches [2]. Learning grasping directly from self-supervision offers considerable promise in this field: if a robot can become progressively better at grasping through repeated experience, perhaps it can achieve a very high degree of proficiency with minimal human involvement. Indeed, learning-based methods inspired by techniques in computer vision have achieved good results in recent years [22]. However, these methods typically do not reason about the sequential aspect of the grasping task, either choosing a single grasp pose [33], or repeatedly choosing the next most promising grasp greedily [24]. While previous works have explored deep reinforcement learning (RL) as a framework for robotic grasping in a sequential decision making context, such studies have been limited to either single objects [34], or simple geometric shapes such as cubes [40].

In this work, we explore how RL can be used to automatically learn robotic grasping skills for diverse objects, with a focus on comparing a variety of RL methods in a realistic simulated benchmark. One of the most important challenges in learning-based grasping is generalization: can the system learn grasping patterns and cues that allow it to succeed at grasping new objects that were not seen during training? Successful generalization typically requires training on a large variety of objects and scenes, so as to acquire generalizeable perception and control. Prior work on supervised learning of grasping has used tens of thousands [33] to millions [24] of grasps, with hundreds of different objects. This regime poses a major challenge for RL: if the learning is conducted primarily on-policy, the robot must repeatedly revisit previously seen objects to avoid forgetting, making it difficult to handle extremely diverse grasping scenarios. Off-policy reinforcement learning methods might therefore be preferred for tasks such as grasping, where the wide variety of previously seen objects is crucial for generalization. Indeed, the supervised learning methods explored in previous work [33], [24] can be formalized as special cases of off-policy reinforcement learning that do not consider the sequential nature of the grasping task.

Our aim in this paper is to understand which off-policy RL algorithms are best suited for vision-based robotic grasping. A number of model-free, off-policy deep reinforcement learning methods have been proposed in recent years for solving tasks such as Atari games [28] and control of simple simulated robots [25]. However, these works do not explore the kinds of diverse and highly varied situations that arise in robotic grasping, and the focus is typically on final performance (e.g., expected reward), rather than generalization to new objects and situations. Furthermore, training typically involves progressively collecting more and more on-policy data, while retaining old off-policy data in a replay buffer. We study how the relative performance of these algorithms varies in an off-policy regime that emphasizes diversity and generalization.

The first contribution of this paper is a simulated grasping benchmark for a robotic arm with a two-finger parallel jaw gripper, grasping random objects from a bin. This task is available as an open-source Gym environment[2] [3]. Next, we present an empirical evaluation of off-policy deep RL algorithms on vision-based robotic grasping tasks. These methods include the grasp success prediction approach proposed by [24], Q-learning [28], path consistency learning (PCL) [29], deep deterministic policy gradient (DDPG) [25],

* Equal contribution

[1]Accompanying video: https://goo.gl/pyMd6p

[2]Code for the grasping environment is available at https://goo.gl/jAESt9

Monte Carlo policy evaluation [39], and Corrected Monte-Carlo, a novel off-policy algorithm that extends Monte Carlo policy evaluation for unbiased off-policy learning.

Our discussion of these methods provide a unified treatment of the various Q-function estimation techniques in the literature, including our novel proposed approach. Our results show that deep RL can successfully learn grasping of diverse objects from raw pixels, and can grasp previously unseen objects in our simulator with an average success rate of 90%. Surprisingly, naïve Monte Carlo evaluation is a strong baseline in this challenging domain, despite being biased in the off-policy case, and our proposed unbiased, corrected version achieves comparable performance. Deep Q-learning also excels in limited data regimes. We also analyze the stability of the different methods, and differences in performance across on-policy and off-policy cases and different amounts of off-policy data. Our results shed light on how the different methods compare on a realistic simulated robotic task, and suggest avenues for developing new, more effective deep RL algorithms for robotic manipulation, discussed in Section VII. To our knowledge, our paper is the first to provide an open benchmark for robotic grasping from image observations and held-out test objects, as well as a detailed comparison of a wide variety of deep RL methods on these tasks.

II. Related Work

A number of works combine RL algorithms with deep neural network function approximators. Model-free algorithms for deep RL generally fall into one of two areas: policy gradient methods [44], [38], [27], [45] and value-based methods [35], [28], [25], [15], [16], with actor-critic algorithms combining the two classes [29], [31], [14]. It is generally well known that model-free deep RL algorithms can be unstable and difficult to tune [18]. Most of the prior works in this field, including popular benchmarks [7], [1], [3], have primarily focused on applications in video games and relatively simple simulated robot locomotion tasks, and do not generally evaluate on diverse tasks that emphasize the need for generalization to new situations. The goal of this work is to evaluate which approaches are suitable for vision-based robotic grasping, in terms of both stability and generalization performance, two factors that are rarely evaluated in standard RL benchmarks.

A number of approaches have sought to apply deep RL methods for solving tasks on real robots. For example, guided policy search methods have been applied for solving a range of manipulation tasks, including contact-rich, vision-based skills [23], non-prehensile manipulation [10], and tasks involving significant discontinuities [5], [4]. Other papers have directly applied model-free algorithms like fitted Q-iteration [21], Monte Carlo return estimates [37], deep deterministic policy gradient [13], trust-region policy optimization [11], and deep Q-networks [46] for learning skills on real robots. These papers have provided excellent examples of successful deep RL applications, but generally tackle individual skills, and do not emphasize generalizing to task instances beyond what the robot was trained on. The goal of this work is to provide a systematic comparison of deep RL approaches to robotic grasping. In particular, we test generalization to new objects in a cluttered environment where objects may be obscured and the environment dynamics are complex, in contrast to works such as [40], [34], and [19], which consider grasping simple geometric shapes such as blocks.

Outside of deep RL, learning policies for grasping diverse sets of objects has been studied extensively in the literature. For a complete survey of approaches, we refer readers to Bohg et al. [2]. Prior methods have typically relied on one of three sources of supervision: human labels [17], [22], geometric criteria for grasp success computed offline [12], and robot self-supervision, measuring grasp success using sensors on the robot's gripper [33]. Deep learning has been recently incorporated into such systems [20], [22], [24], [26], [32]. These prior methods do not consider the sequential decision making formalism of grasping maneuvers, whereas our focus in this paper is on evaluating RL algorithms for grasping. We do include a comparison to a prior method that learns to predict grasp outcomes without considering the sequential nature of the task [24], and observe that deep RL methods are more suitable in harder, more cluttered environments.

Finally, a primary consideration of this paper is the ability to effectively learn from large amounts of off-policy data, which makes deploying new algorithms much more practical. Sadheghi et al. use deep reinforcement learning to from offline simulated data to learn a model for drone flight [37]. Other papers have considered large-scale data collection for robotics. For example, Finn et al. learn a predictive model of sensory inputs and used it to plan [8], [9]. Pinto & Gupta [33] and Levine et al. [24] both use supervised learning techniques for learning to grasp. Unlike these prior approaches, we focus on model-free RL algorithms, which can consider the future consequences of their actions (e.g., in order to enable pregrasp manipulation).

III. Preliminaries

We first define the RL problem and the notation that we use in the rest of the paper. We consider a finite-horizon, discounted Markov decision process (MDP): at each timestep t, the agent will observe the current state $\mathbf{s}_t \in S$, take an action $\mathbf{a}_t \in A$, and then receive a reward $r(\mathbf{s}_t, \mathbf{a}_t)$ and observe the next state $\mathbf{s}_{t+1}$, each stochastically determined by the environment. Episodes have length T timesteps. The goal of the agent is to find a policy $\mathbf{a} \sim \pi_\theta(-|\mathbf{s})$, parameterized by θ, under which the expected reward is maximized. We will additionally assume that future rewards are discounted by γ, such that the objective becomes:

$$\max_\theta \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\pi_\theta}\left[\sum_{t=1}^{T} \gamma^{t-1} r(\mathbf{s}_t, \mathbf{a}_t)\right]. \quad (1)$$

Note that the expectation is with respect to both the policy and the environment dynamics. To reduce notational clutter, we use $\mathbb{E}$ without a subscript to refer to expectation only

over the environment dynamics (and not a specific policy) and specify a specific policy only when relevant.

IV. Problem Setup

Our proposed benchmark for vision-based robotic grasping build on top of the Bullet simulator [6]. In this environment, a robotic arm with 7 degrees of freedom attempts to grasp objects from a bin. The arm has a fixed number of timesteps ($T = 15$) to find a good grasp, at which point the gripper closes and the episode ends. The reward is binary and provided only at the last step, with $r(\mathbf{s}_T, \mathbf{a}_T) = 1$ for a successful grasp and 0 for a failed grasp. The observed state $\mathbf{s}_t$ consists of the current RGB image from the viewpoint of the robot's camera and the current timestep t (Figure 1). The timestep is included in the state, since the policy must know how many steps remain in the episode to decide whether, for example, it has time for a pre-grasp manipulation, or whether it must immediately move into a good grasping position. The arm moves via position control of the vertically-oriented gripper. Continuous actions are represented by a Cartesian displacement $[dx, dy, dz, d\phi]$, where ϕ is a rotation of the wrist around the z-axis. The gripper automatically closes when it moves below a fixed height threshold, and the episode ends. At the beginning of each new episode, the object positions and rotations are randomized within the bin.

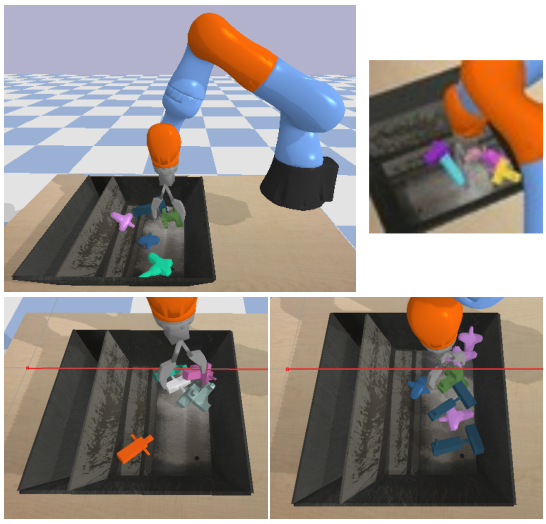

Fig. 1. Upper Left: An illustration of our simulated grasping setup. The robot must pick up objects in a bin, which is populated using randomized objects shown in Figure 2. Upper right: Example observations to the robot. Bottom left: In the first task, the robot picks up a wide variety of randomized objects and generalizes to unseen test objects. Bottom right: In the second task, the robot has to pick one of the purple cross-shaped objects from a cluttered bin.

(a) (b)

Fig. 2. Left: 30 of the 900 train objects. Right: 10 of the 100 test objects.

The benchmark consists of two different RL environments, shown in in Figure 1.

1) **Regular grasping.** The first grasping task tests generalization, with 900 randomly generated rigid objects with diverse random shapes used during training, and 100 testing performed on new objects on which the model was never trained previously. In each episode there are 5 objects in the bin. Every 20 episodes, the objects are randomly switched out. Both training and test objects are visualized in Figure 2.
2) **Targeted grasping in clutter.** In this task, the robot must pick up a particular cross-shaped object in a bin with many other objects, which may occlude each other visually (See Figure 1 right). The arm may disturb other objects in the bin when attempting to select and grasp the "target object". We chose this setting because grasping specific objects in clutter may require more nuanced behavior from the robot. The robot trains on objects which are kept the same for all episodes. We evaluate performance on sets of 7 objects where 3 of them are "target" objects, and the robot only receives reward for picking up one of the target objects.

V. Reinforcement Learning Algorithms

In addition to proposing a vision-based grasping benchmark, we aim to evaluate off-policy deep RL algorithms to determine which methods are best suited for learning complex robotic manipulation skills, such as grasping, in diverse settings that require generalization to novel objects. Our detailed experimental evaluation includes well-known algorithms such as Q-learning [42], [41], deep deterministic policy gradient (DDPG) [25], which we show to be a variant of Q-learning with approximate maximization, path consistency learning (PCL) [29], Monte Carlo policy evaluation [39], which consists of simple supervised regression onto estimated returns, and a novel corrected version of Monte Carlo policy evaluation, which makes the algorithm unbiased in the off-policy case, with a correction term that resembles Q-learning and PCL.

A. Learning to Grasp with Supervised Learning

The first method in our comparison is based on the grasping controller described by Levine et al. [24]. This method does not consider long-horizon returns, but instead uses a greedy controller to choose the actions with the highest predicted probability of producing a successful grasp. We include this approach in our comparison because it is a recent example of a prior grasping method that learns to perform closed-loop feedback control using deep neural networks from raw monocular images. To our knowledge, no prior method learns vision-based robotic grasping with deep networks for grasping of diverse objects with reinforcement learning, making this prior approach the closest point of comparison.

This prior method learns an outcome predictor $Q_\theta(\mathbf{s}, \mathbf{a})$ for the next-step reward after taking a single action $\mathbf{a}$. This amounts to learning a single-step Q-function. To obtain

labeled data from multi-step grasping episodes, this method uses "synthetic" actions obtained by taking the position of the gripper at any point during the episode, denoted $\mathbf{p}_t$, and computing the action that would move the gripper to the final pose of the episode $\mathbf{p}_T$. Since actions correspond to changes in gripper pose, the action label is simply given by $\mathbf{a}_t = \mathbf{p}_T - \mathbf{p}_t$, and the outcome of the entire episode is used as the label for each step within that episode. This introduces bias: taking a straight-line path from $\mathbf{p}_t$ to $\mathbf{p}_T$ does not always produce the same grasp outcome as the actual sequence of intermediate steps taken in the corresponding episode. The action $\mathbf{a}_t$ is selected by maximizing the Q-function $Q_\theta(\mathbf{s},\mathbf{a})$ via stochastic optimization. In our implementation, we employ the cross-entropy method (CEM), with 3 iterations and 64 samples per iteration. For further details, we refer the reader to prior work [24].

B. Off-Policy Q-Learning

We begin by describing the standard off-policy Q-learning algorithm [42], which is one of the best known and most popular methods in this class. Q-learning aims to estimate the Q-function by minimizing the Bellman error, given by

$$\mathscr{E} = \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a}}\left[\left(Q_\theta(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + \gamma \max_{\mathbf{a}'} Q_\theta(\mathbf{s}',\mathbf{a}'))\right)^2\right]. \tag{2}$$

Expectations over $\mathbf{s},\mathbf{a}$ correspond to state-action pairs sampled from an off-policy replay buffer. Minimizing this quantity for all states results in the optimal Q-function, which induces an optimal policy. In practice, the Bellman error is minimized only at sampled states, by computing the gradient of Equation (2) with respect to the Q-function parameters θ and using stochastic gradient descent. The gradient is computed only through the $Q_\theta(\mathbf{s},\mathbf{a})$ term, without considering the derivative of the non-differentiable max operator. Applying Q-learning off-policy is then straightforward: batches of states, actions, rewards, and subsequent states, of the form $(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1},r_t)$ are sampled from the buffer of stored transition tuples, and the gradient of the Bellman error is computed on these samples. In practice, a number of modifications to this method are employed for stability, as suggested in prior work [28]. First, we employ a *target network* inside the max that is decorrelated from the learned Q-function, by keeping a lagged copy of the Q-function that is delayed by 50 gradient updates. We refer to this target network as $Q_{\theta'}$. Second, we employ double Q-learning (DQL) [41], which we found in practice improves the performance of this algorithm. In double Q-learning, the max operator uses the action that maximizes the current network Q_θ, but the value obtained from the target network $Q_{\theta'}$, resulting in the following error estimate:

$$\mathscr{E} = \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a}}\left[\left(Q_\theta(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + \gamma Q_{\theta'}(\mathbf{s}', \arg\max_{\mathbf{a}'} Q_\theta(\mathbf{s}',\mathbf{a}')))\right)^2\right].$$

To handle continuous actions, we use a simple stochastic optimization method to compute the arg max in the target value: we sample 16 actions uniformly at random, and pick the one with the largest Q-value. While this method is crude, it is efficient, easy to parallelize, and we found it to work well for our 4-dimensional action parameterization. The action at execution time is selected with CEM, in the same way as described in the previous section.

C. Regression with Monte Carlo Return Estimates

Although the Q-learning algorithm discussed in the previous section is one of the most commonly used and popular Q-function learning methods for deep reinforcement learning, it is far from the simplest. In fact, if we can collect on-policy data, we can estimate Q-values directly with supervised regression. Assuming episodes of length T, the empirical loss for Monte Carlo policy evaluation [39] is given by

$$\mathscr{E} = \frac{1}{2}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(Q_\theta(\mathbf{s}_t,\mathbf{a}_t) - \sum_{t'=t}^{T}\gamma^{t'-t} r(\mathbf{s}_t,\mathbf{a}_t)\right)^2,$$

where the first sum is taken over sampled episodes, and the second sum over the time steps within each episode. If the samples are drawn from the latest policy, this method provides an unbiased approach to estimating Q-values. Monte Carlo return estimates were previously used to learn deep reinforcement learning polices for drone flight in [37]. In contrast to Q-learning, it does not require bootstrapping – the use of the most recent function approximator or target network to estimate target values. This makes the method very simple and stable, since the optimization reduces completely to standard supervised regression. However, the requirement to obtain on-policy samples severely limits the applicability of this approach for real-world robotic manipulation. In our experiments, we evaluate how well this kind of Q-function estimator performs when employed on off-policy data. Surprisingly, it provides a very competitive alternative, despite being a biased estimator in the off-policy case.

D. Corrected Monte Carlo Evaluation

The Monte Carlo (MC) policy evaluation algorithm described in the previous section is a well-known method for estimating Q-values [39], but not an especially popular one: it does not benefit from bootstrapping, and is biased when applied in an off-policy setting. We can improve this approach by removing the off-policy bias through the addition of a correction term, which we describe in this section. This correction is a novel contribution of our paper, motivated by the surprising effectiveness of the naïve Monte Carlo evaluation method. Let Q^* and V^* be the Q-values and state values of the optimal policy:

$$Q^*(\mathbf{s}_t,\mathbf{a}_t) = \mathbb{E}_{\mathbf{s},\mathbf{a}}[r(\mathbf{s}_t,\mathbf{a}_t) + \gamma V^*(\mathbf{s}_{t+1})], \tag{3}$$

$$V^*(\mathbf{s}_t) = \max_{\mathbf{a}} Q^*(\mathbf{s}_t,\mathbf{a}). \tag{4}$$

We may express the advantage of a state-action pair as

$$A^*(\mathbf{s}_t,\mathbf{a}_t) = Q^*(\mathbf{s}_t,\mathbf{a}_t) - V^*(\mathbf{s}_t) \tag{5}$$

$$= \mathbb{E}_{\mathbf{s},\mathbf{a}}[r(\mathbf{s}_t,\mathbf{a}_t) + \gamma V^*(\mathbf{s}_{t+1}) - V^*(\mathbf{s}_t)]. \tag{6}$$

Thus we have

$$\mathbb{E}_{\mathbf{s},\mathbf{a}}[V^*(\mathbf{s}_t) - \gamma V^*(\mathbf{s}_{t+1})] = \mathbb{E}_{\mathbf{s},\mathbf{a}}[r(\mathbf{s}_t,\mathbf{a}_t) - A^*(\mathbf{s}_t,\mathbf{a}_t)]. \tag{7}$$

If we perform a discounted sum of the two sides of Equation 7 over $\mathbf{s}_t, \ldots, \mathbf{s}_T$ we induce a telescoping cancellation:

$$\mathbb{E}_{\mathbf{s},\mathbf{a}}\left[\sum_{t'=t}^{T}\gamma^{t'-t}(V^*(\mathbf{s}_{t'})-\gamma V^*(\mathbf{s}_{t'+1}))\right] = \mathbb{E}_{\mathbf{s},\mathbf{a}}\left[\sum_{t'=t}^{T}\gamma^{t'-t}(r(\mathbf{s}_{t'},a_{t'})-A^*(\mathbf{s}_{t'},\mathbf{a}_{t'}))\right] \quad (8)$$

$$\Rightarrow V^*(\mathbf{s}_t) = \mathbb{E}_{\mathbf{s},\mathbf{a}}\left[\sum_{t'=t}^{T}\gamma^{t'-t}(r(\mathbf{s}_{t'},\mathbf{a}_{t'})-A^*(\mathbf{s}_{t'},\mathbf{a}_{t'}))\right], \quad (9)$$

where we recall that $V^*(\mathbf{s}_{T+1}) = 0$. Equivalently, we have

$$Q^*(\mathbf{s}_t,\mathbf{a}_t) = \mathbb{E}_{\mathbf{s},\mathbf{a}}\left[r(\mathbf{s}_t,\mathbf{a}_t)+\sum_{t'=t+1}^{T}\gamma^{t'-t}(r(\mathbf{s}_{t'},\mathbf{a}_{t'})-A^*(\mathbf{s}_{t'},\mathbf{a}_{t'}))\right]. \quad (10)$$

Thus, we may train a parameterized Q_θ to minimize the squared difference between the LHS and RHS of Equation 10. Note that this resulting algorithm is a modified version of Monte Carlo augmented with a correction to the future reward given by the discounted sum of advantages. Another interpretation of this correction is the difference between the Q-values of the actions actually taken along the sampled trajectory, and the optimal actions. This means that "good" actions along "bad" trajectories are given higher values, while "bad" actions along "good" trajectories are given lower values. This removes the bias of Monte Carlo when applied to off-policy data. We also note that this corrected Monte Carlo may be understood as a variant of PCL [29], discussed below, without the entropy regularization. In practice, we also multiply the correction term by a coefficient ν, which we anneal from 0 to 1 during training to improve stability. When $\nu = 0$, the method corresponds to supervised regression, and when $\nu = 1$, it becomes unbiased.

E. Deep Deterministic Policy Gradient

Deep deterministic policy gradient (DDPG) [25] is an algorithm that combines elements of Q-learning and policy gradients. Originally derived from the theory of deterministic policy gradients, this algorithm aims to learn a deterministic policy $\pi_\phi(\mathbf{s}) = \mathbf{a}$, by propagating gradients through a critic $Q_\theta(\mathbf{s},\mathbf{a})$. However, DDPG can also be interpreted as an approximate Q-learning algorithm. To see this, observe that, in Q-learning, the policy that is used at test time is obtained by solving $\pi^*(\mathbf{s}) = \arg\max_{\mathbf{a}} Q_\theta(\mathbf{s},\mathbf{a})$. In continuous action spaces, performing this optimization at every decision step is computationally expensive. The actor, which is trained in DDPG according to the objective

$$\max_{\phi}\mathbb{E}_{\mathbf{s},\mathbf{a}}\left[Q_\theta(\mathbf{s},\pi_\phi(\mathbf{s}))\right], \quad (11)$$

can be seen as an approximate maximizer of the Q-function with respect to the action at any given state $\mathbf{s}$. This amortizes the search over actions. The update equations in DDPG closely resemble Q-learning. The Q-function is updated according to the gradient of the bootstrapped objective

$$\mathscr{E} = \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a}}\left[\left(Q_\theta(\mathbf{s},\mathbf{a})-(r(\mathbf{s},\mathbf{a})+\gamma Q_{\theta'}(\mathbf{s}',\pi_\phi(\mathbf{s}'))\right)^2\right],$$

and the actor is updated by taking one gradient step for the maximization in Equation (11). Comparing this equation to that of standard double Q-learning, we see that the only difference for the Q-function update is the use of $\pi_\phi(\mathbf{s}')$ instead of the arg max. The practical implementation of DDPG closely follows that of Q-learning, with the addition of the actor update step after each Q-function update. In practice, a lagged ("target network") version of the actor is used to compute the target value [25].

F. Path Consistency Learning

Path consistency learning (PCL) [29] is a stochastic optimal control variant of Q-learning that resembles our corrected Monte Carlo method. Though the full derivation of this algorithm is outside the scope of this paper, we briefly summarize its implementation, and include it for comparison due to its similarity with corrected MC. PCL augments the RL objective in Equation (1) with a τ-weighted discounted entropy regularizer,

$$\pi^* = \arg\max_{\pi_\theta}\ \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\pi_\theta}\left[\sum_{t=1}^{T}\gamma^{t-1}(r(\mathbf{s}_t,\mathbf{a}_t)-\tau\log\pi_\theta(\mathbf{a}_t|\mathbf{s}_t))\right].$$

The corresponding optimal value function is given by

$$V^*(\mathbf{s}_t) = \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\pi^*}\left[\sum_{t'=t}^{T}\gamma^{t'-t}(r(\mathbf{s}_{t'},\mathbf{a}_{t'})-\tau\log\pi^*(\mathbf{a}_{t'}|\mathbf{s}_{t'}))\right],$$

and together the policy and value function must satisfy d-step consistency for any $d > 0$:

$$V^*(\mathbf{s}_t) = \mathbb{E}_{\mathbf{s},\mathbf{a}}\left[\gamma^d V^*(\mathbf{s}_{t+d})+\sum_{t'=t}^{t+d-1}\gamma^{t'-t}(r(\mathbf{s}_{t'},\mathbf{a}_{t'})-\tau\log\pi^*(\mathbf{a}_{t'}|\mathbf{s}_{t'}))\right]. \quad (12)$$

PCL minimizes the squared difference between the LHS and RHS of Equation 12 for a parameterized π_θ, V_ϕ. In our experiments, we use a variant Trust-PCL [30], which uses a Gaussian policy and modifies the entropy regularizer to relative entropy with respect to a prior version of the policy.

G. Summary and Unified View

In this section, we provide a unified view that summarizes the individual choices made in each of the above algorithms. All of the methods perform regression onto some kind of target value to estimate a Q-function, and the principal distinguishing factors among these methods consist of the following two choices:

a) Bootstrapping or Monte Carlo returns: The standard Q-learning algorithm and DDPG use the bootstrap, by employing the current function approximator Q_θ (or, in practice, a target network $Q_{\theta'}$) to determine the value of the policy at the next step, via the term $\max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}')$. In contrast, both Monte Carlo variants, PCL, and the single-step supervised method use the actual return of the entire episode. This is in general biased in the off-policy case, since the current policy might perform better than the policy that collected the data. In the case of Monte Carlo with corrections and PCL, a

correction term is added to compensate for the bias. We will see that adding the correction to Monte Carlo substantially improves performance.

b) Maximization via an actor, or via sampling: The DDPG and PCL methods use a second network to choose the actions, while the other algorithms only learn a Q-function, and choose actions by maximizing it with stochastic search. The use of a separate actor network has considerable benefits: obtaining the action from the actor is much faster than stochastic search, and the actor training process can have an amortizing effect that can accelerate learning [25]. However, our empirical experiments show that this comes at a price: learning a value function and its corresponding actor function jointly makes them co-dependent on each other's output distribution, resulting in instability.

The following table summarizes the specific choices for these two parameters made by each of the algorithms:

algorithm	target value	action selection
supervised learning	episode value[3]	stochastic search
Q-learning	bootstrapped	stochastic search
Monte Carlo	episode value	stochastic search
Corrected Monte Carlo	corrected episode value	stochastic search
DDPG	bootstrapped	actor network
PCL	corrected episode value[4]	actor network

VI. Experiments

We evaluate each RL algorithm along four axes: overall performance, data-efficiency, robustness to off-policy data, and hyperparameter sensitivity, all of which are important for the practicality of applying these methods to real robotic systems. As discussed in Section IV, we consider two challenging simulated grasping scenarios, regular and targeted grasping, with performance evaluated on held-out test objects. All algorithms use variants of the deep neural network architecture shown in Figure 3 to represent the Q-function.

A. Data Efficiency and Performance

We consider learning with both on-policy and off-policy data. In each setting we initialize the pool of experience with an amount of random-policy data (10k, 100k, or 1M grasps)[5]. A Q-function model is trained from this data. In the on-policy case we periodically sample 50 on-policy grasps every 1k training steps which are used to augment the initial pool. This setting is on-policy in the sense that we continually recollect data with the latest policy. However, the amount of on-policy data is still significantly less than traditional on-policy algorithms which sample a batch of on-policy experience for each gradient step. This procedure is thus more representative of a robotic learning setting, where data collection is much more expensive than training iterations. We find that the difference between off-policy and on-policy is slight across

[3]Supervised learning uses the actual episode value, but does not use the actual actions of the episode, instead merging multiple actions into a cumulative action that leads to the episode's final state.

[4]PCL also includes an entropy regularizer in the objective.

[5]Code for the random policy is available at https://goo.gl/hPS6ca

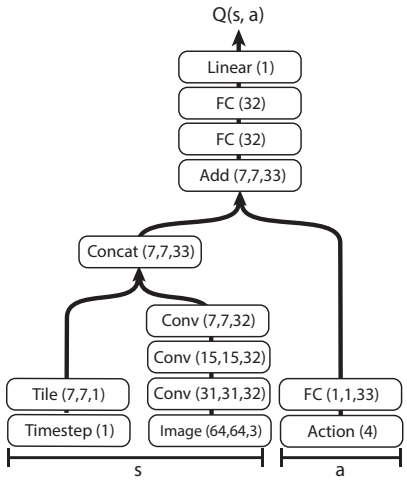

Fig. 3. Q network architecture used for grasping tasks, with layer sizes in parentheses. The model takes as input the timestep **t**, the image observation **s**, and candidate action **a**.

all algorithms in all environments (see Figures 4-5). This suggests that the amount of on-policy data necessary to have a significant benefit of performance is more than what we applied, and therefore likely unfeasible for robotics.

Overall, DQL, supervised learning, MC, and our Corr-MC variant learn the most successful policies given enough data (see Figures 4-5). DQL tends to perform better in low-data regimes, while MC and corrected MC achieve slightly better performance in the high-data regime on the harder targeted task. The good performance of DQL in low-data regimes can be partially explained by the variance reduction effect of the bootstrapped target estimate.

Although Corr-MC and standard MC perform well, often competitively with DQL in high-data regimes, standard MC does not actually perform substantially worse than Corr-MC in most cases. Although standard MC is highly biased in the off-policy setting, it still achieves good results, except in the lowest data regimes with purely off-policy data. This suggests that the bias incurred from this approach may not be as disastrous as generally believed, which should merit further investigation in future work. It is clear that while supervised training can perform well, standard model-free deep RL methods can perform competitively and, in some cases, slightly better. Generally, DDPG and PCL perform poorly compared to the other baselines.

B. Analyzing Stability

When applying deep RL methods to real robotic systems, we care not only about performance and data efficiency, but robustness to different hyperparameter values. Extensively tuning a learning algorithm for a particular environment can be tedious and impractical, and current deep RL methods are known to be unstable [18]. In this section, we will study the robustness of each algorithm to hyperparameters and different random seeds. For each algorithm we sweep over different values for learning rate (0.01, 0.001, 0.0001), number of convolution filters and fully-connected units in

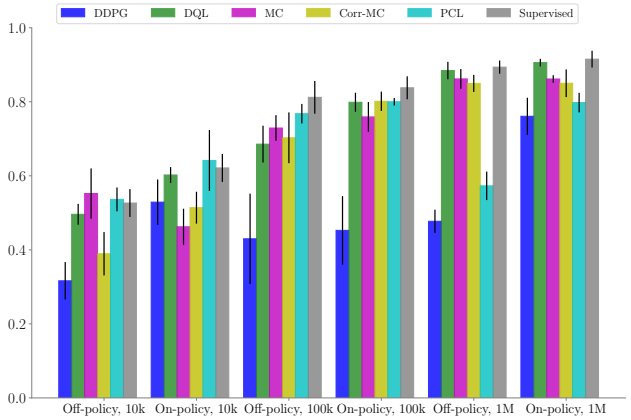

Fig. 4. Regular grasping performance on held-out test objects for varying dataset sizes. DQL and the supervised baseline perform best. Standard deviations computed from 9 independent runs with different random seeds.

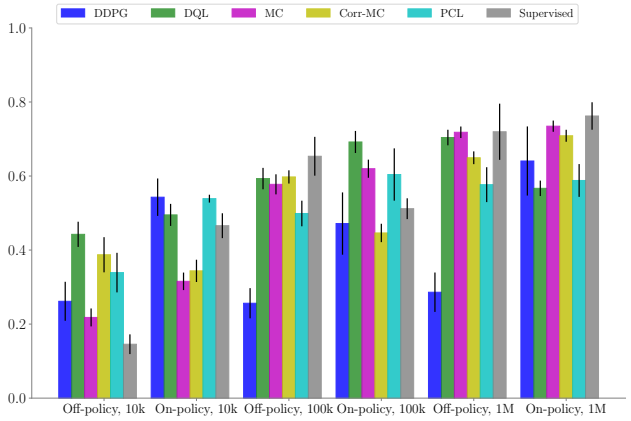

Fig. 5. Targeted grasping performance in a cluttered bin with three target object and four non-target objects, for varying dataset sizes. DQL performs well in the low-data and off-policy regimes, whereas MC and corrected-MC performs best with maximal data.

each layer (32, 64), discount factor (0.9, 0.95)[6], and duration (in training steps) of per-step exploration with a linearly decaying schedule (10000, 10000). All hyperparameter sweeps were done in the on-policy learning setting with 100k initial random grasps.

In Figure 6, we show an analysis of the sensitivity of each algorithm to each combination of the aforementioned hyperparameter values and 9 random seeds. Our results show that DQL, Corr-MC, PCL, MC, and Supervised are relatively stable across different hyperparameter values. This plot is insightful, showing that although MC and Corr-MC yield similar performance given optimal hyperparameters, the unbiased Corr-MC is slightly more robust to hyperparameter choice. The performance of DDPG drops substantially for suboptimal hyperparameters. Correspondingly, DDPG (which is the least stable) typically achieves the worst performance in our experiments. These results strongly indicate that algorithms that employ a second network for the actor suffer a considerable drop in stability, while approximate maximization via stochastic search, though crude, provides

[6]MC and Supervised do not use the discount factor hyperparameter.

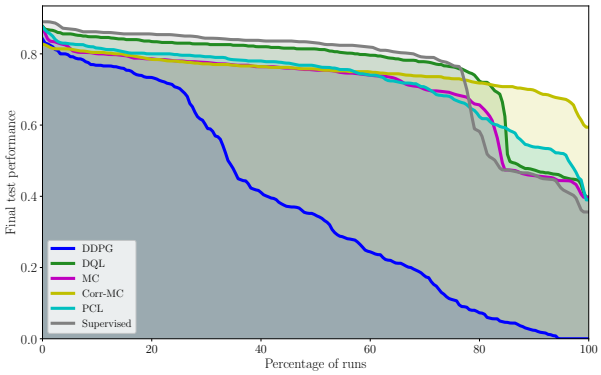

Fig. 6. Grasping success rate on held-out test objects for every hyperparameter setting in our sweep, sorted in decreasing order. DQL, PCL, and Corr-MC methods are relatively stable, while DDPG is comparatively unstable.

significant benefits in this regard.

VII. Discussion and Future Work

We presented an empirical evaluation of a range of off-policy, model-free deep reinforcement learning algorithms. Our set of algorithms includes popular model-free methods such as double Q-learning, DDPG, and PCL, as well as a prior method based on supervised learning with synthetic actions [24]. We also include a naïve Monte Carlo method, which is biased in the off-policy case but surprisingly achieves reasonable performance, often outperforming DDPG, and a corrected version of this Monte Carlo method, which is a novel contribution of this work. Our experiments are conducted in a diverse grasping simulator on two types of tasks: a grasping task that evaluates generalization to novel random objects not seen during training, and a targeted grasping task that requires isolating and grasping a particular type of object in clutter.

Our evaluation indicates that DQL performs better on both grasping tasks than other algorithms in low-data regimes, for both off-policy and on-policy learning, and additionally having the desirable property of being relatively robust to choice of hyperparameters. When data is more plentiful, algorithms that regress to a multistep return, such as Monte Carlo or the corrected variant of Monte Carlo typically achieve slightly better performance. When considering the algorithm features summarized in Section V-G, we find that the use of an actor network substantially reduces stability, leading to poor performance and severe hyperparameter sensitivity. Methods that use entire episode values for supervision tend to perform somewhat better when data is plentiful, while the bootstrapped DQL method performs substantially better in low data regimes. These insights suggest that, in robotic settings where off-policy data is available, single-network methods may be preferred for stability, and methods that use (corrected) full episode returns should be preferred when data is plentiful, while bootstrapped methods are better in low data regimes. A natural implication of this result is that future research into robotic reinforcement learning algorithms might focus on combining the best of bootstrapping and

multistep returns, by adjusting the type of target value based on data availability. Another natural extension of our work is to evaluate a similar range of methods in real-world settings. Since the algorithms we evaluate all operate successfully in off-policy regimes, they are likely to be reasonably practical to use in realistic settings.

VIII. Acknowledgements

We thank Laura Downs, Erwin Coumans, Ethan Holly, John-Michael Burke, and Peter Pastor for helping with experiments.

References

[1] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research (JAIR)*, 2013.

[2] J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-driven grasp synthesisa survey. *Transactions on Robotics*, 2014.

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

[4] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. *International Conference on Machine Learning (ICML)*, 2017.

[5] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine. Path integral guided policy search. In *International Conference on Robotics and Automation (ICRA)*, 2017.

[6] E. Coumans and Y. Bai. pybullet, a python module for physics simulation, games, robotics and machine learning. http://pybullet.org/, 2016–2017.

[7] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning (ICML)*, 2016.

[8] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *Neural Information Processing Systems (NIPS)*, 2016.

[9] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *International Conference on Robotics and Automation (ICRA)*, 2017.

[10] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *International Conference on Robotics and Automation (ICRA)*, 2016.

[11] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Björkman. Deep predictive policy training using reinforcement learning. *arXiv preprint*, 2017.

[12] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen. The columbia grasp database. In *International Conference on Robotics and Automation (ICRA)*, 2009.

[13] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. *International Conference on Robotics and Automation (ICRA)*, 2017.

[14] S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *International Conference on Learning Representations (ICLR)*, 2017.

[15] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep Q-learning with model-based acceleration. In *International Conference on Machine Learning (ICML)*, 2016.

[16] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. *International Conference on Machine Learning (ICML)*, 2017.

[17] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, J. Bohg, T. Asfour, and S. Schaal. Learning of grasp selection based on shape-templates. *Autonomous Robots*, 2014.

[18] R. Islam, P. Henderson, M. Gomrokchi, and D. Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint*, 2017.

[19] S. James and E. Johns. 3d simulation for robot arm control with deep q-learning. *arXiv preprint*, 2016.

[20] D. Kappler, J. Bohg, and S. Schaal. Leveraging big data for grasp planning. In *International Conference on Robotics and Automation (ICRA)*, 2015.

[21] S. Lange, M. Riedmiller, and A. Voigtlander. Autonomous reinforcement learning on raw visual input data in a real world application. In *International Joint Conference on Neural Networks (IJCNN)*, 2012.

[22] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research (IJRR)*, 2015.

[23] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research (JMLR)*, 2016.

[24] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research (IJRR)*, 2016.

[25] 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. *International Conference on Learning Representations (ICLR)*, 2016.

[26] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint*, 2017.

[27] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint*, 2013.

[29] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Neural Information Processing Systems (NIPS)*, 2017.

[30] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans. Trust-pcl: An off-policy trust region method for continuous control. *arXiv preprint*, 2017.

[31] B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih. PGQ: Combining policy gradient and Q-learning. *arXiv preprint*, 2016.

[32] A. t. Pas, M. Gualtieri, K. Saenko, and R. Platt. Grasp pose detection in point clouds. *arXiv preprint*, 2017.

[33] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *International Conference on Robotics and Automation (ICRA)*, 2016.

[34] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint*, 2017.

[35] M. Riedmiller. Neural fitted q iteration-first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning (ECML)*. Springer, 2005.

[36] A. Rodriguez, M. T. Mason, and S. Ferry. From caging to grasping. *International Journal of Robotics Research (IJRR)*, 2012.

[37] F. Sadeghi and S. Levine. (cad)$ˆ2$rl: Real single-image flight without a single real image. *CoRR*, abs/1611.04201, 2016.

[38] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pages 1889–1897, 2015.

[39] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[40] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint*, 2017.

[41] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. *AAAI*, 2016.

[42] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 1992.

[43] J. Weisz and P. K. Allen. Pose error robust grasping from contact wrench space metrics. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 557–562. IEEE, 2012.

[44] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

[45] Y. Wu, E. Mansimov, S. Liao, R. Grosse, and J. Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *arXiv preprint*, 2017.

[46] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint*, 2015.