# Off-Policy Evaluation of Generalization for Deep Q-Learning in Binary Reward Tasks

**Alex Irpan** [1] **Kanishka Rao** [1] **Konstantinos Bousmalis** [2] **Chris Harris** [1] **Julian Ibarz** [1] **Sergey Levine** [1 3]

## Abstract

In the off-policy policy evaluation (OPE) problem, we want to estimate an agent's performance without online interaction with the environment, which is difficult due to out-of-distribution problems between the learned agent and the validation set of offline interactions. OPE is commonly done through importance weighting or model learning. In this paper, we propose an alternative OPE metric, focusing on the special (but relatively common) case of deterministic MDPs with sparse binary rewards, that uses a learned critic $Q(s, a)$ as a classifier, using its accuracy to estimate return of the corresponding policy. Due to only requiring a Q-function estimate, we can apply the proposed metric to learning regimes where importance sampling or model fitting is difficult or infeasible. Experiments in toy and Atari environments show the metric correlates with return better than ad hoc approaches like the TD error. Turning an eye towards cross-domain generalization, we test the OPE metric on a difficult high-dimensional image-based real-world robot grasping setup. When applied to models trained only in simulation, the metric continues to correlate well with returns, even when the testing environment is in the real-world and uses objects not seen at training time. This opens the potential to heavily reduce real-robot usage when developing new models.

## 1. Introduction

Supervised learning has seen significant advances due to the emergence of large labeled datasets (Deng et al., 2009), and the use of validation sets to estimate their final performance on the true test distribution, even if that test distribution differs from the training set. The analogue in reinforcement learning (RL) is off-policy policy evaluation (OPE), where we predict the performance of a learned RL policy using validation data collected by a different, fixed policy. This is unnecessary for simulated environments, but becomes important in real-world settings where online interaction is risky, costly, or time consuming (Thomas et al., 2015), since a robust off-policy metric can reject poor policies without risking harm in the real-world. In settings where we aim to transfer a policy from a simulated environment to a more complex one, such as robot grasping from simulation to reality (Bousmalis et al., 2018), OPE could let us perform model selection and algorithm design entirely in simulation, reducing robot management overhead.

Previous approaches for off-policy policy evaluation (OPE) (Precup et al., 2000; Dudik et al., 2011; Jiang & Li, 2015; Thomas & Brunskill, 2016) use a mix of importance sampling and model learning to evaluate policies on their validation sets. This can prove challenging for value-based RL methods. Such methods may learn deterministic policies (Lillicrap et al., 2015; van Hasselt et al., 2016), which have undefined importance weights for off-policy actions. The training data may include human expert demonstrations (Večerík et al., 2017), the action distribution of which is not explicitly known. In high-dimensional and continuous state spaces, importance weights can degenerate to zero (Levine & Koltun, 2013), making them tricky to work with. As for model-based techniques, although there has been progress on high-dimensional modelling (Babaeizadeh et al., 2018; Lee et al., 2018), learning sufficiently accurate models in image space for effective evaluation is still an open research problem.

Seeking a more general approach, we observe that almost all RL algorithms learn a Q-function, either through Q-learning or as part of an actor-critic algorithm. Although general MDPs can have complex dynamics and reward functions, many tasks have deterministic dynamics and can be defined by a simple binary reward function, where reward indicates success or failure at a task. Focusing on this case, we derive an OPE metric using only a learned $Q(\mathbf{s}, \mathbf{a})$ and an off-policy validation set. Our method does not use importance weighting and does not assume knowledge of which behavior policy collected the validation trajectories.

[1]Google Brain, Mountain View, USA [2]DeepMind, London, UK [3]University of California Berkeley, Berkeley, USA. Correspondence to: Alex Irpan <alexirpan@google.com>.

*Proceedings of the RL for Real Life International Conference on Machine Learning 2019 Workshop*, Long Beach, California. 

At its core, our approach is based on the observation that in deterministic binary reward tasks, the actions at every state can be partitioned into "good" and "bad" actions, and return can be related to whether the policy chooses good actions or not. We present a theoretical motivation of our approach based on semi-supervised classification. Our analysis is based on deterministic environments, but we show strong results in practice across several environments, including a non-deterministic real-world robotic grasping task on unseen test objects (Kalashnikov et al., 2018).

## 2. Preliminaries

We focus on finite–horizon, deterministic Markov decision processes (MDP). We define an MDP as $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{S}_0, r, \gamma)$. $\mathcal{S}$ is the state–space, $\mathcal{A}$ the action–space, and both can be continuous. $\mathcal{P}$ defines transitions to next states given current state and action, $\mathcal{S}_0$ defines initial state distribution, $r$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. We assume the MDP is undiscounted ($\gamma = 1$). Episodes are of finite length $T$: at a given time-step $t$ the agent is at state $\mathbf{s}_t \in \mathcal{S}$, samples an action $\mathbf{a}_t \in \mathcal{A}$ from a policy $\pi$, receives a reward $\mathbf{r}_t = r(\mathbf{s}_t, \mathbf{a}_t)$, and observes the next state $\mathbf{s}_{t+1}$ as determined by $\mathcal{P}$. Trajectories $(\mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \mathbf{a}_2, \ldots)$ are represented by $\tau$, and $R(\pi)$ is expected episode return.

The goal in RL is to learn a policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ that maximizes the return $R(\pi_\theta) = \mathbb{E}_{\boldsymbol{s},\boldsymbol{a}}[\sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$. A value of a policy for given state $\mathbf{s}_t$ is defined as $V^\pi(\mathbf{s}_t) = \mathbb{E}_\pi[\sum_{t'=t}^{T} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}^\pi_{t'})]$ where $\mathbf{a}^\pi_t$ is the action $\pi$ takes at state $\mathbf{s}_t$ and $\mathbb{E}_\pi$ implies an expectation over trajectories sampled from $\pi$. Given a policy $\pi$, the expected value of its action $a_t$ at a state $s_t$, is called the Q-value and is defined as $Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_\pi[r(\mathbf{s}_t, \mathbf{a}_t) + \sum_{t'=t+1}^{T} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}^\pi_{t'})]$.

We refer to it as *off-policy* when a policy, $\pi$, is being trained or evaluated on a data-set $\mathcal{D}$ consisting of episodes generated by a different policy $\pi_b$. In contrast, *on-policy* training is when the learned policy, $\pi$, itself generates the episodes. In off-policy evaluation, learned policies $\{\pi_1, \pi_2, \cdots\}$ are evaluated against a fixed $\mathcal{D}$ generated by some unknown $\pi_b$.

### 2.1. Deep Q-Learning

In Q-learning, we learn a policy $\pi$ by training a Q-function, approximated by a neural network with parameters $\theta$, to minimize the average TD error.

$$\mathcal{E} = \left(Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - r(\mathbf{s}_t, \mathbf{a}_t) - \gamma \max_{\mathbf{a}} Q^\pi(\mathbf{s}_{t+1}, \mathbf{a})\right)^2 \tag{1}$$

We then define $\pi(\mathbf{s}_t) = \arg\max_a Q^\pi(\mathbf{s}_t, \mathbf{a}; \theta)$. In continuous action spaces, we may approximate $\max_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a})$ by using the cross-entropy method (CEM), as done in Quillen et al. (2018) and Kalashnikov et al. (2018). $Q^\pi$ can be trained with on-policy interaction, or fit offline to a fixed $\mathcal{D}$ with batch Q-learning. The learned $\pi$ is then deployed to a target environment that may differ from the training environment. In this work, we treat the RL algorithm as a black-box that produces learned Q-functions $\hat{Q}(\mathbf{s}, \mathbf{a})$, and seek to evaluate $\pi(\mathbf{s}) = \arg\max_{\mathbf{a}} \hat{Q}(\mathbf{s}, \mathbf{a})$.

## 3. Off-Policy Evaluation via Classification

Formally, a deterministic binary reward MDP satisfies the following properties: transitions $\mathcal{P}$ are deterministic, reward $\mathbf{r}_t = 0$ at all intermediate steps, and final reward $\mathbf{r}_t$ lies in $\{0, 1\}$, indicating whether the final state is a failure or success. In such an MDP, given $(\mathbf{s}_t, \mathbf{a}_t)$, we say $\mathbf{a}_t$ is **good** if the optimal policy $\pi^*$ can achieve +1 return if initialized at $\mathbf{s}_{t+1}$. Equivalently, there exists a sequence of future actions that reaches a success state. Action $\mathbf{a}_t$ is **bad** if no such sequence exists. Under this definition, the return of trajectory $\tau$ is 1 if and only if all $\mathbf{a}_t$ are good (see Appendix A.1), letting us view return as a sequence of classification problems.

**Theorem 1.** *Given binary reward MDP $\mathcal{M}$ and policy $\pi$, let $\rho_{t,\pi,good}(\mathbf{s})$ denote the state distribution $\pi$ visits at time $t$, given that actions $\mathbf{a}_1, \cdots, \mathbf{a}_{t-1}$ were all good. Let $\mathcal{A}_-(\mathbf{s})$ denote the set of bad actions at state $\mathbf{s}$, and let $\epsilon_t = \mathbb{E}_{\rho_{t,\pi,good}}\left[\sum_{\mathbf{a}\in\mathcal{A}_-(\mathbf{s})} \pi(\mathbf{a}|\mathbf{s})\right]$ be the per-step expectation of $\pi$ making its first mistake at time $t$, with $\epsilon = 1/T \sum_{i=1}^{T} \epsilon_t$ the average of $\epsilon_t$ over time. Then $R(\pi) \geq 1 - T\epsilon$, and this lower bound is tight.*

This is proved by using behavioral cloning bounds from imitation learning, proved by (Ross & Bagnell, 2010), using properties of binary reward MDPs to improve the cost bound from $O(T^2\epsilon)$ to $O(T\epsilon)$. See Appendix A for details.

A smaller $\epsilon$ gives a higher lower bound on return, which implies a better $\pi$. This leaves estimating $\epsilon$ from the $\mathcal{D}$ collected by $\pi_b$. There are two difficulties to doing so. One is that our validation dataset is from $\pi_b$, whose state distribution does not match the $\rho_{t,\pi,good}$ distribution that defines $\epsilon_t$. The other challenge is that we do not have negative labels, so we cannot directly access to $\mathcal{A}_-(\mathbf{s})$.

### 3.1. Distribution Mismatch Between $\pi_b$ And $\rho_{t,\pi,good}$

Theroem 1 relies on estimating $\epsilon$ against distribution $\rho_{t,\pi,good}$ from dataset $\mathcal{D}$ from $\pi_b$. A natural approach would be importance weighting $(\mathbf{s}, \mathbf{a})$ appropriately, but we assume no knowledge of $\pi_b$, and this is not well-defined for deterministic policies $\pi(\mathbf{s}) = \arg\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$. Another approach is to subsample $\mathcal{D}$ to transitions $(\mathbf{s}, \mathbf{a})$ where $\mathbf{a} = \pi(\mathbf{s})$. This ensures an on-policy evaluation, and (Liu et al., 2018) have found success with this approach, but this can hit finite sample issues if $\pi_b$ does not sample $\pi(\mathbf{s})$ frequently enough. This is especially unlikely in continuous

action spaces.

Therefore, we assume estimating classification accuracy over $\mathcal{D}$ is sufficient to estimate classification over $\{(\mathbf{s}, \pi(\mathbf{s})) : \mathbf{s} \in \mathcal{S}\}$. This is admittedly a strong assumption, but empirical results in Section 6 show surprising robustness to distributional mismatch. We expect this assumption to become worse for environments with long time horizons $T$, where it is easier for $\pi$ and $\pi_b$ to visit very different states.

### 3.2. Missing Negative Labels

Recall $(\mathbf{s}_t, \mathbf{a}_t)$ is good if $\pi^*$ can succeed from $\mathbf{s}_{t+1}$, and bad otherwise. Since $\pi^*$ is at least as good as $\pi_b$, whenever $\pi_b$ succeeds, all $(\mathbf{s}_t, \mathbf{a}_t)$ in trajectory $\tau$ must be good. However, the converse is not true, since $\pi^*$ could have succeeded from $(\mathbf{s}_t, \mathbf{a}_t)$ where $\pi_b$ failed. This is an instance of the positive-unlabeled (PU) learning problem. We briefly summarize PU learning, following the treatment from (Kiryo et al., 2017).

Let $(X, Y)$ be a labeled binary classification problem, with $Y \in \{\pm 1\}$. Let $g : X \to \mathbb{R}$ be some decision function, and let $\ell : \mathbb{R} \times \{\pm 1\} \to \mathbb{R}$ be our loss function. In the RL setting, $X = \mathcal{S} \times \mathcal{A}$, labels $\{\pm 1\} = \{good, bad\}$, and a natural choice for $g$ is $g(\mathbf{s}, \mathbf{a}) = Q(\mathbf{s}, \mathbf{a})$.

We want to estimate $\epsilon$, the probability $\pi$ takes a bad action i.e. generating a false positive. This is not the risk $\mathcal{R}(g)$, since $\mathcal{R}(g)$ accounts for both false-positives and false-negatives. If $(\mathbf{s}, \pi(\mathbf{s}))$ is predicted to be bad, but is actually good, it does not impact reward - what matters is the ground truth label. We want to bound just the false-positive risk.

Let $\pi_p, \pi_n$ be the class priors $\pi_p = P(Y = +1), \pi_n = P(Y = -1)$. The false positive risk $\mathcal{R}_{fp}(g)$ of classifier $g$ is the risk over just negative examples.

$$\epsilon = \mathcal{R}_{fp}(g) = \pi_n \mathbb{E}_{X|Y=-1} \left[\ell(g(X), -1)\right] \tag{2}$$

This can be estimated indirectly from positive labels. Since $\pi_n p(x|y = -1) = p(x) - \pi_p p(x|y = +1)$, for any $f(x)$, $\pi_n \mathbb{E}_{X|Y=-1} \left[f(X)\right] = \mathbb{E}_X \left[f(X)\right] - \pi_p \mathbb{E}_{X|Y=+1} \left[f(X)\right]$. Let $f(x) = \ell(g(x), -1)$ to estimate $\mathcal{R}_{fp}$.

$$\epsilon \approx \mathbb{E}_{X,Y} \left[\ell(g(X), -1)\right] - \pi_p \mathbb{E}_{X|Y=+1} \left[\ell(g(X), -1)\right] \tag{3}$$

Eqn. 3 is the core of all metrics this paper proposes, and experiments are done with different choices of $\ell$ and $\pi_p$.

### 3.3. Off-Policy Classification Score

The Off-Policy Classification (OPC) score is defined by the negative loss when $\ell$ in Eqn. 3 is the 0-1 loss. Let $b$ be a threshold, with $\ell(Q(s, a), Y) = \frac{1}{2}\left(1 - Y \operatorname{sign}(Q(s, a) - b)\right)$. This gives

$$OPC(Q) = \pi_p \mathbb{E}_{pos} \left[1_{Q(s,a)>b}\right] - \mathbb{E}_{all} \left[1_{Q(s,a)>b}\right] \tag{4}$$

To ensure independence from the magnitude of $Q(\mathbf{s}, \mathbf{a})$, $b$ is fit separately for each Q-function to maximize $OPC(Q)$. Given $N$ transitions and $Q(\mathbf{s}, \mathbf{a})$ for all $(\mathbf{s}, \mathbf{a}) \in \mathcal{D}$, the best $b$ for each $Q$ is computable in $O(N \log N)$ time. See Appendix B for details.

### 3.4. Utilizing the OPC Score

We can use the OPC score to comparatively evaluate different models using a validation set $\mathcal{D}$ according to the following procedure, which we test in our experiments:

- Collect data $\mathcal{D}$ using a behavior policy $\pi_b$. For every successful trajectory $\tau$, label all $(\mathbf{s}, \mathbf{a})$ as positive. This $\mathcal{D}$ is the validation dataset and is fixed for all models.
- Train several Q-functions $Q^\pi$ with different RL algorithms. The algorithm should estimate $Q^\pi(\mathbf{s}, \mathbf{a})$, using $\pi(\mathbf{s}) = \arg\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$ as the final eval policy. This is compatible with Q-learning and deterministic actor-critic approaches, and can be applied to both batch Q-learning and online Q-learning.
- For each $Q^\pi$, evaluate $Q^\pi$ over $\mathcal{D}$ using Eqn. 3, then select the $Q^\pi$ with best score. One implementation detail is that in our experiments, we have episodes of different length. To avoid focusing on long episodes, transitions $(\mathbf{s}, \mathbf{a})$ from an episode of length $T$ are weighted by $1/T$ when estimating $\epsilon$.

### 3.5. SoftOPC Score

Alternatively, $\ell$ can be a soft loss function. We experimented with $\ell(Q(s, a), Y) = -YQ(s, a)$, which is minimized when $Q(s, a)$ is large for $Y = +1$ and small for $Y = -1$. The negative of this loss is called the SoftOPC.

$$SoftOPC(Q) = \pi_p \mathbb{E}_{pos} \left[Q(s, a)\right] - \mathbb{E}_{all} \left[Q(s, a)\right] \tag{5}$$

### 3.6. Baseline Metrics

Since we assume importance-sampling and model learning is infeasible, many common OPE baselines do not fit our problem setting. As alternatives, we derive several Q-learning based metrics, which also assume no knowledge of $\pi_b$ and only requires a $Q(\mathbf{s}, \mathbf{a})$ estimate. In all baselines, $\mathbf{a}_t^\pi$ is the on-policy action $\arg\max_{\mathbf{a}} Q^\pi(\mathbf{s}_t, \mathbf{a})$.

**Temporal-Difference Error** A natural baseline is to use the Q-learning training loss (the TD error) as the evaluation metric. We compute the average TD error over $\mathcal{D}$.

$$\mathbb{E}_{\pi_b} \left[\left(Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - (\mathbf{r}_t + \gamma Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_t^\pi))\right)^2\right] \tag{6}$$

**Discounted Sum of Advantages** The difference of the value functions of two policies $\pi$ and $\pi_b$ at state $\mathbf{s}_t$ is given

by the discounted sum of advantages (Kakade & Langford, 2002; Murphy, 2005) of $\pi$ on episodes induced by $\pi_b$:

$$V^{\pi_b}(\mathbf{s}_t) - V^{\pi}(\mathbf{s}_t) = \mathbb{E}_{\pi_b}\left[\sum_{t'=t}^{T} \gamma^{t'-t} A^{\pi}(\mathbf{s}_{t'}, \mathbf{a}_{t'})\right], \quad (7)$$

where $\gamma$ is the discount factor and $A^{\pi}$ the advantage function for policy $\pi$, defined as $A^{\pi}(\mathbf{s}_t, \mathbf{a_t}) = Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) - Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t^{\pi})$. Since $V^{\pi_b}$ is fixed, estimating (7) is sufficient to compare $\pi_1$ and $\pi_2$. The $\pi$ with smaller score is better.

$$\mathbb{E}_{\mathbf{s}_t \sim \pi_b}\left[\sum_{t'=t}^{T} \gamma^{t'-t} A^{\pi}(\mathbf{s}_{t'}, \mathbf{a}_{t'})\right]. \quad (8)$$

**Monte-Carlo Estimate Corrected with the Discounted Sum of Advantages** Estimating $V^{\pi_b}(\mathbf{s}_t) = \mathbb{E}_{\pi_b}[\sum_{t'} \gamma^{t'-t} \mathbf{r}_{t'}]$ with the Monte-Carlo return, substituting into Eqn. (7), and rearranging gives

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

With $V^{\pi}(\mathbf{s}_t) + A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$, we can obtain an approximate $\widetilde{Q}$ estimate depending on the whole episode:

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

The MCC Error is the squared error to this estimate.

$$\mathbb{E}_{\pi_b}\left(Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) - \widetilde{Q}_{MCC}(\mathbf{s}_t, \mathbf{a}_t, \pi)\right)^2 \quad (11)$$

Note that (11) was proposed before by Quillen et al. (2018) as a training loss for a Q-learning variant, but not for the purposes of off-policy evaluation.

Eqn. (6) and Eqn. (10) share the same optimal Q-function, so assuming a perfect learning algorithm, there is no difference in information between these metrics. In practice, the Q-function will not be perfect due to errors in function approximation and optimization. Eqn. (10) is designed to rely on all future rewards from time $t$, rather than just $\mathbf{r}_t$. We theorized that using more of the "ground truth" from $\mathcal{D}$ could improve the metric's performance in imperfect learning scenarios.

Each of these baselines is a different way to measure how well $Q(\mathbf{s}, \mathbf{a})$ fits the data. However, it is possible to learn a good $\pi$ even when $Q(\mathbf{s}, \mathbf{a})$ poorly fits the data. If $\pi$ is defined as $\arg\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$, then $Q^*(\mathbf{s}, \mathbf{a})$ will produce $\pi^*$ and have 0 TD error, but many other Q-functions will also produce $\pi^*$ with non-zero TD error. This motivates the classification metrics: since $\pi$'s behavior only depends on the relative differences in Q-value, it makes sense to use metrics defined by contrasting Q-values against each other, rather than metrics based on error between the Q-values and episode return. Figure 1 visualizes the differences between the proposed metrics and baseline metrics.

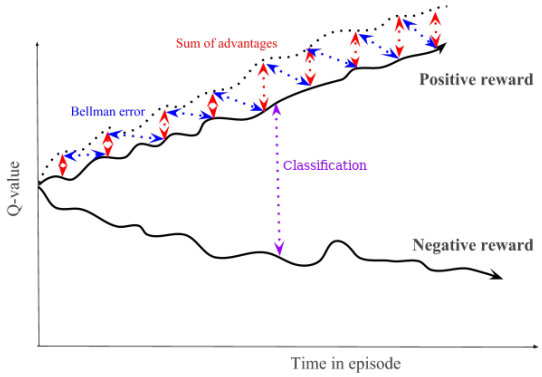


*Figure 1.* **A visual summary of the off-policy metrics.** The two solid curves represent $Q(\mathbf{s}, \mathbf{a})$ over two trajectories in the off-policy data-set that succeed or fail. The dashed curve represents the learned policy's $Q(\mathbf{s}, \mathbf{a}^{\pi})$ along the $\mathbf{s}$ the positive trajectory visits (the corresponding negative trajectory curves are omitted for simplicity). Visually, the TD error can be interpreted as the average *horizontal* difference, along the same $(\mathbf{s}, \mathbf{a})$. The sum of advantages is the sum of *vertical* differences along the same time. The classification metrics directly measure separation between success and failure trajectories.

## 4. Related Work

Off-policy policy evaluation (OPE) predicts return of a learned policy $\pi_e$ from a fixed set of off-policy data $\mathcal{D}$, generated by one or more behavior policies $\pi_b$. Prior works (Precup et al., 2000; Dudik et al., 2011; Jiang & Li, 2015; Liu et al., 2018; Theocharous et al., 2015; Hanna et al., 2017) do so with importance sampling (IS) (Horvitz & Thompson, 1952), MDP modeling, or both.

Doing IS requires querying $\pi_e(\mathbf{a}|\mathbf{s})$ and $\pi_b(\mathbf{a}|\mathbf{s})$ for any $\mathbf{s} \in \mathcal{D}$, to correct for the shift in state-action distributions. In RL, the cumulative product of IS weights along $\tau$ is used to weight its contribution to $\pi_e$'s estimated value (Precup et al., 2000). Several variants have been proposed, such as step-wise IS and weighted IS (Mahmood et al., 2014).

In MDP modeling, a model is fit to $\mathcal{D}$, and $\pi_e$ is rolled out in the learned model to estimate average return (Mannor et al., 2007; Jiang & Li, 2015). The performance of these approaches is worse if dynamics or reward are poorly estimated, which tends to occur for image-based tasks. Improving these models is an active research question in deep RL (Babaeizadeh et al., 2018; Lee et al., 2018).

IS-based estimators and model-based estimators can be combined with established statistical approaches (Cassel et al., 1976) to produce an improved unbiased estimator. These doubly robust estimators have been applied to bandits (Dudik et al., 2011; Dudík et al., 2014) and the sequential setting (Jiang & Li, 2015; Thomas & Brunskill, 2016), and can be combined with bootstrap estimates of a model ensemble to give empirical bounds (Hanna et al., 2017).

This paper aims to study OPE on a large image-based task, with a continuous action space, where the environment shifts at test-time. To our knowledge, no other OPE papers have proposed approaches applicable when both importance sampling and model learning are difficult or infeasible.

## 5. Applications of OPE to Generalization

The fundamental goal of off-policy policy evaluation is about estimating performance in one distribution using data from another. This is very similar to the problem of generalization, where we wish to estimate performance in train and test environments. In this section, we discuss ways OPE can be applied to measure generalization, and its importance for doing so in real-world environments.

Reinforcement learning (RL) typically uses the same environment for both training and testing (Mnih et al., 2015), but recent work has discussed issues with this paradigm by examining overfitting and memorization in deep reinforcement learning. Several papers (Zhang et al., 2018b; Raghu et al., 2018; Cobbe et al., 2018; Zhang et al., 2018a) have shown deep RL agents can memorize input levels, then fail to generalize to test-set levels. These approaches are not directly related to off-policy evaluation, but share many of the same difficulties: the distribution mismatch between training data and the final environment makes it hard to extrapolate training environment performance to test environment performance. New RL benchmarks with explicit train-test environment splits have been proposed to spur research into generalization for deep RL (Nichol et al., 2018; Cobbe et al., 2018).

However, simply defining a test environment does not by itself provide a useful mechanism for model evaluation. Recent work focuses on problems defined in simulation, where it is easy to evaluate the policy in the test environment. When developing for a real-world setting, where test environment evaluation is expensive, off-policy evaluation is an inescapable part of measuring generalization performance in an efficient, tractable way.

### 5.1. Scenarios for Off-Policy Evaluation

We identify some common overfitting scenarios faced in reinforcement learning, in which we can evaluate off-policy metrics to measure generalization.

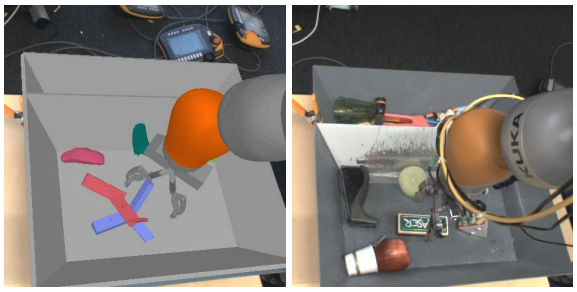

*Figure 2.* **Robotics grasping simulation-reality gap.** The state RGB image shown for simulation (left) and reality (right).

1. **Simulation-to-reality:** Models trained on-policy in simulation may not generalize to the real-world, since simulators are not perfect and can have a considerable reality-gap (see Figure 2). Policies can over-fit to the simulation environment, exploiting their inaccuracies and failing to generalize to the real-world without transfer learning techniques (James et al., 2018).
2. **Insufficient off-policy training data:** Models trained with off-policy data (collected by a behavior policy $\pi_b$) may over-fit if there is insufficient training data. Without extra interaction, the model may memorize the state-action pairs in the training data and fail to generalize to new state-action pairs. This is rarely a concern in standard RL benchmarks, since in these benchmarks RL algorithms collect new on-policy data with high enough frequency. However, choices in how to store and replay old experience can impact performance in ways that are hard to predict without policy evaluation (Zhang & Sutton, 2017; Liu & Zou, 2017).
3. **Mismatched off-policy training data:** Even if the model has arbitrary amounts of off-policy training data, the behavior policy may miss important regions of the state space, making it possible for the model to fail to capture the target distribution. There is enough data for the model to avoid memorizing training trajectories, but the learned model can still overfit to overarching regularities. For example, a robotics grasping model trained with arbitrary amounts of data on a fixed set objects may not generalize to a test set of objects.

All of these scenarios are, in principle, identifiable by off-policy evaluation, as long as validation is done against an appropriate sample of data sampled in the final testing environment. We evaluate metrics across all such regimes.

## 6. Experiments

### 6.1. Binary Tree Environment

An open question in the metric is how to choose the positive class prior $\pi_p$. We examine these questions in a toy binary

tree environment. The environment is a full binary tree of $k = 6$ levels, where each node is a state, and actions are {left,right}. Leaf nodes are terminal with reward 0 or 1. The initial state distribution is uniform over all non-leaf nodes. The validation dataset is collected by the uniformly random policy. OPE metrics are commonly evaluated by their MSE to episode return, but since we are not directly estimating episode return, we instead use correlation between the OPE metric and episode return as our final score.

We generate 1000 random Q-functions where $Q(\mathbf{s}, \mathbf{a}) \sim U[0, 1]$, define $\pi(\mathbf{s}) = \arg\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$, then plot Spearman correlation for the SoftOPC and OPC for different $\pi_p$. Spearman correlation $\rho$ is used over Pearson correlation $r$ to measure rank consistency. Plots are generated over two extremes: only 1 leaf is a fail state, or only 1 leaf is a success state. Figure 3 shows the SoftOPC outperforms OPC and both outperform all baselines. Interestingly, prior $\pi_p = 1$ performs best, even when almost all states are fail states. One theory is that $\pi_p$ should be picked as the upper bound of the return $R_{\pi^*}$ of the optimal policy on states that can reach a positive outcome, which is always 1. See Appendix C for details. We use $\pi_p = 1$ in all future experiments.

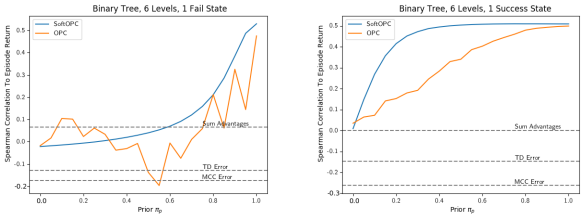


*Figure 3.* Spearman correlation of SoftOPC, OPC, and baselines with varying $\pi_p$. Baselines do not depend on $\pi_p$. Correlations further from 0 are better. Interestingly, it is easier to predict performance when almost all trajectories lead to failure.

### 6.2. Atari Results

For Atari experiments, we chose Pong as our environment, since it is well-studied environment that can be changed to a binary reward env by truncating the episode after either player fails to return the ball. The final reward is $-1$ or $+1$.

We train models using DQN (Mnih et al., 2015) and DDQN (van Hasselt et al., 2016), varying hyperparameters like learning rate, $\gamma$, and batch size. The validation dataset is formed by taking the 38 DDQN checkpoints and generating 30 episodes from each, giving 1140 episodes total. Measuring both Pearson correlation and Spearman correlation, we find the OPC outperforms the SoftOPC, with $R^2 = 0.499, \rho = 0.72$, compared to $R^2 = 0.363, \rho = 0.75$. We attribute this to the 0-1 property of the OPC score. In Figure 4, models trained with $\gamma = 0.9$ instead of $\gamma = 0.99$ are highlighted. Q-values for these models lies in a much smaller region, and are hard to separate with the soft loss. The 0-1 loss in OPC separates the Q-values for these models more easily.

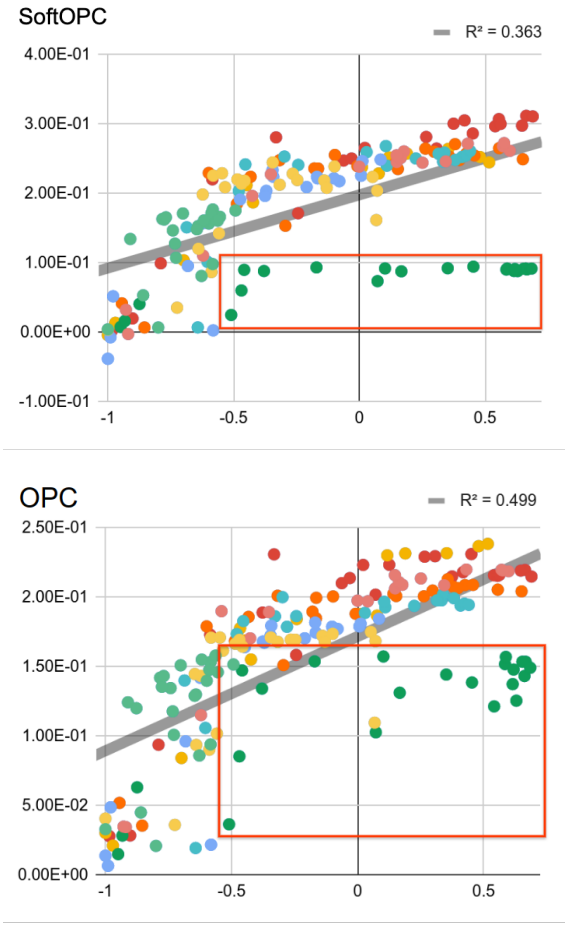


*Figure 4.* Scatterplot of episode return (x-axis) against metric (y-axis), for SoftOPC (top) and OPC (bottom). Each color is a different hyperparameter setting (explained in Appendix D). Green points within the red rectangle are DQN, $\gamma = 0.9$ models, and are better separated by OPC.

## 7. Generalization Experiments

To evaluate generalization performance in a large-scale task, we use simulated and real versions of a vision-based robotic grasping task from (Kalashnikov et al., 2018). The task is briefly summarized below.

The observation at each time-step is a 640 x 512 RGB image of the robot and a bin of objects, cropped to 472 x 472 before inference. The goal is to grasp any of the objects in that bin. Actions are continuous moves of the gripper in space. Reward is 0 except at the final timestep, where $r(\mathbf{s}_T, \mathbf{a}_T) = 1$ on a successful grasp and 0 otherwise. All models in this section use the QT-Opt algorithm (Kalashnikov et al., 2018)

*Table 1.* Summarized results of Experiments section. The $R^2$ of line of best fit and Spearman correlation $\rho$ for each metric (leftmost column) is evaluated in each environment (top row). These are: the toy tree MDP from Section 6.1, Pong from Section 6.2, simulated grasping with train or target objects from Section 7.1, and real-world grasping from Section 7.2. Occasionally, some baselines correlate well, but the proposed metrics (last two rows) are consistently among the best metrics for each environment. All results use $\pi_p = 1$.

| | **Tree (1 Fail)** | | **Tree (1 Succ)** | | **Pong** | | **Sim Train** | | **Sim Target** | | **Real-World** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $R^2$ | $\rho$ | $R^2$ | $\rho$ | $R^2$ | $\rho$ | $R^2$ | $\rho$ | $R^2$ | $\rho$ | $R^2$ | $\rho$ |
| **TD Error** | 0.006 | -0.129 | 0.016 | -0.147 | 0.048 | -0.175 | 0.022 | -0.372 | 0.105 | -0.514 | 0.17 | 0.48 |
| $\sum \gamma^t A^\pi$ | 0.003 | 0.065 | 0.000 | 0.001 | 0.091 | -0.319 | **0.735** | 0.813 | **0.744** | **0.782** | 0.12 | 0.50 |
| **MCC Error** | 0.021 | -0.175 | 0.063 | -0.263 | 0.035 | -0.355 | 0.000 | 0.331 | 0.058 | -0.436 | 0.01 | -0.15 |
| **OPC** | 0.207 | 0.475 | **0.210** | 0.499 | **0.499** | 0.720 | 0.490 | **0.861** | 0.350 | 0.660 | 0.81 | 0.87 |
| **SoftOPC** | **0.229** | **0.530** | 0.195 | **0.509** | 0.363 | **0.750** | 0.553 | 0.759 | 0.476 | 0.768 | **0.91** | **0.94** |

with the same architecture and hyperparameters.

### 7.1. Simulation experiments

Per our discussion in Section 5.1, a model may overfit because of: *(a)* insufficient amounts of off-policy training data, or, *(b)* because of a distribution mismatch between training and test data. We first show we can induce both. Two sets of 5 random objects are sampled. One object set is used for training (Sim Train), and the other is used for testing (Sim Target). The goal is to generalize to these target objects.[1].

Simulation models are trained with batch Q-learning without on-policy interaction with the environment. To generate a dataset for batch Q-learning, we collected 950,000 simulated grasps from a human-designed policy (60% success rate) with $\epsilon$-greedy exploration ($\epsilon = 0.1$). 50,000 of these grasps were set aside as a dedicated holdout set, leaving 900,000 training grasps. The same policy was used to collect 10,000 grasps on the target objects, as a validation set.

We show *insufficient amounts of off-policy training data* and *mismatched off-policy training data*. In Figure 5, we trained two models with a limited 100k grasps dataset or a large 900k grasps dataset, then evaluated grasp success on the same objects it was trained on. We see the the model overfits to the limited dataset. Comparing TD error, we also see a clear difference between the validation set (same objects) and the target set (target objects), showing overfitting to the training environment. With overfitting reproduced, we evaluate which metrics track final performance most reliably for both scenarios.

Following Section 6.2, we measure $R^2$ and $\rho$ over all models from both the 100k and 900k grasp training runs, giving $n = 1144$ models total. Correlation and episode return is estimated over both the training objects and the target objects. Models from the first 100,000 steps of training are excluded, since the Q-function heavily underfits the data at the start of training, leading to outliers in all metrics.

We see that SoftOPC and OPC perform similarly and beat

[1]Models taken from https://sites.google.com/site/brainrobotdata/home/models.

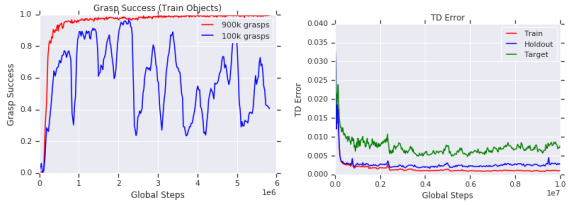


*Figure 5.* **Overfitting to dataset and environment** *Top:* Grasp success curve of model trained with 900k or 100k grasps. Model lacking data oscillates in performance, even for objects the model sees at train time. *Bottom:* TD error of the 100k grasp model on offline data for training, holdout data for validation, and test data on unseen target objects. As the model memorizes the training set, train and holdout curves separate. Target object TD error is consistently higher than both.

several baselines, and their correlation holds for the target objects. The best metric is the discounted sum of advantages. Plots of their performance can be seen in Figure 6 and Figure 7. Oddly, the discounted sum of advantages correlates well in the opposite of the direction expected by theory. It is unclear why this occurred, but one possibility is that more negative advantages indicate Q-function overconfidence, rather than true improvement in action selection.

### 7.2. Simulation-to-Reality Evaluation

For our final experiments, we evaluate simulation overfitting and generalization to the real-world. We examine 15 different models, trained to have varying degrees of robustness to the simulation-to-reality gap. These models are described in detail in (James et al., 2018), which proposes Randomized-to-Canonical Adaptation Networks (RCANs) to alleviate the simulation-to-reality gap. Among the 15 Q-functions are models trained purely in simulation, models trained with domain randomization, models trained with RCAN, and models trained with real robot data. These $Q(\mathbf{s}, \mathbf{a})$ cover a wide range of real-world performance, with real-world grasp success ranging from 17% to 91%, making them a good real-world testbed. A useful off-policy metric should predict real-world grasp success without additional real-world interaction.

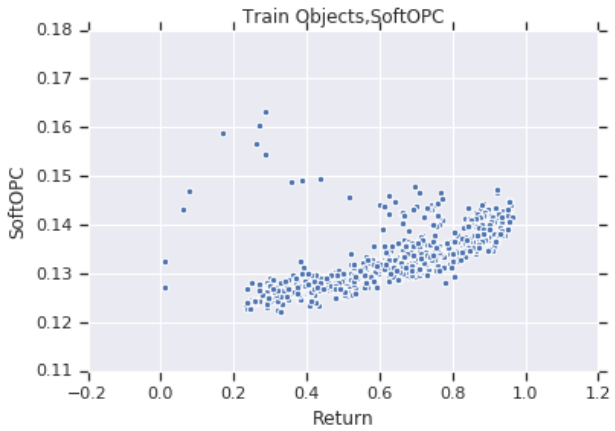


*Figure 6.* Scatterplot of SoftOPC in Sim Train environment. Each data point is a different model checkpoint, corresponding to (return, metric value).

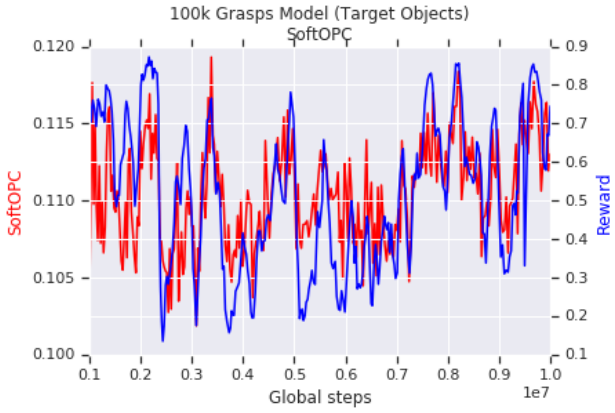


*Figure 7.* Overlay of SoftOPC (red) and return (blue) for Sim Target environment. Plot excludes the first $10^6$ steps of training, since SoftOPC is more stable after the Q-function has partially converged. SoftOPC tracks peaks and valleys in episode return.

Real-world grasp success is evaluated with 6 real robots, each grasping different objects. Each of the 6 robots execute 102 grasps for a total of 612 grasps. For the off-policy metrics, an off-policy data-set is collected on the real robots. This dataset had 400,000 grasps, of which roughly 40% are successful. A subsampled dataset is used for validation. Objects used in real-world grasp success evaluation were not in this validation set, combining the test object shift from Section 7.1 with sim-to-real domain shift.

Figure 8 shows scatterplots of each metric and Table 1 summarizes the $R^2$ and $\rho$ of each metric. Again, the SoftOPC and OPC outperformed all baselines, even though real-world dynamics are non-deterministic. The SoftOPC does slightly better. We also see $\sum \gamma^{t'} A^{\pi}(\mathbf{s}_{t'}, \mathbf{a}_{t'})$ correlates poorly, despite results from Section 7.1. To study robustness to the validation dataset used, we also evaluated the SoftOPC on different subsamples of $\mathcal{D}$. Appendix E shows correlation was mostly unchanged.

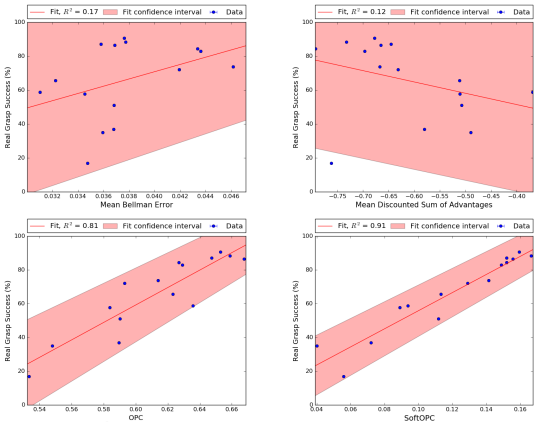


*Figure 8.* **Off-policy metrics versus real-world grasping success.** Left-to-right, top-to-bottom, scatterplots for: TD error, $\sum \gamma^{t'} A^{\pi}(\mathbf{s}_{t'}, \mathbf{a}_{t'})$, OPC, and SoftOPC over a real-world validation set. Each data point is a different grasping model, where y-axis is real-world grasp success. For each metric a least squares fit is performed and the $R^2$ and the 95% confidence intervals are shown. The OPC and SoftOPC are the only good indicators of real-world model performance.

## 8. Conclusion and Future Work

We propose classification-based off-policy evaluation metrics that can be used in any setting where we learn $Q(\mathbf{s}, \mathbf{a})$, which includes settings where existing OPE metrics do not apply. The proposed metrics work with any RL algorithm that learns a Q-function, and can be applied even when that actor is only implicitly defined. The metrics require the special case of a deterministic, binary reward MDP, but this class is wide enough to cover several environments, including real-world robotics tasks. Empirically, we find the SoftOPC and OPC scores correlate well with performance across several environments, and successfully identify cases where Q-functions overfit to their training regime, whether they have too little data or data from the wrong distribution. These approaches successfully correlated with return in the simulation-to-reality scenario, a critical setting for robotics. Effective off-policy evaluation for the real world on models trained in simulation enables development of simulation-based RL algorithms with many fewer real-world evaluations. The robustness of these classification metrics to the dataset used gives increased confidence these approaches can be generally useful. In future work, we would like to evaluate these metrics on other binary-reward tasks, consider extensions to tasks without binary rewards or with stochasticity, and experiment with directly optimizing against the proposed metrics.

## References


Babaeizadeh, M., Finn, C., Erhan, D., Campbell, R. H., and Levine, S. Stochastic variational video prediction. In *International Conference on Representation Learning*, 2018.

Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K., et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4243–4250. IEEE, 2018.

Cassel, C. M., Särndal, C. E., and Wretman, J. H. Some results on generalized difference estimation and generalized regression estimation for finite populations. *Biometrika*, 63(3):615–620, 1976.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR, 2009*, 2009.

Dudik, M., Langford, J., and Li, L. Doubly robust policy evaluation and learning. In *ICML*, March 2011.

Dudík, M., Erhan, D., Langford, J., Li, L., et al. Doubly robust policy evaluation and optimization. *Statistical Science*, 29(4):485–511, 2014.

Hanna, J. P., Stone, P., and Niekum, S. Bootstrapping with models: Confidence intervals for Off-Policy evaluation. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pp. 538–546, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

Horvitz, D. G. and Thompson, D. J. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685, 1952.

James, S., Wohlhart, P., Kalakrishnan, M., Kalashnikov, D., Irpan, A., Ibarz, J., Levine, S., Hadsell, R., and Bousmalis, K. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. 12 2018.

Jiang, N. and Li, L. Doubly robust off-policy value evaluation for reinforcement learning. November 2015.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *ICML*, 2002.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

Kiryo, R., Niu, G., du Plessis, M. C., and Sugiyama, M. Positive-unlabeled learning with non-negative risk estimator. In *NeurIPS*, pp. 1675–1685, 2017.

Lee, A. X., Zhang, R., Ebert, F., Abbeel, P., Finn, C., and Levine, S. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

Levine, S. and Koltun, V. Guided policy search. In *International Conference on Machine Learning*, pp. 1–9, 2013.

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

Liu, R. and Zou, J. The effects of memory replay in reinforcement learning. In *ICML 2017 Reinforcement Learning Workshop*, 2017.

Liu, Y., Gottesman, O., Raghu, A., Komorowski, M., Faisal, A., Doshi-Velez, F., and Brunskill, E. Representation balancing mdps for off-policy policy evaluation. In *NeurIPS*, 2018.

Mahmood, A. R., van Hasselt, H. P., and Sutton, R. S. Weighted importance sampling for off-policy learning with linear function approximation. In *Advances in Neural Information Processing Systems*, pp. 3014–3022, 2014.

Mannor, S., Simester, D., Sun, P., and Tsitsiklis, J. N. Bias and variance approximation in value function estimates. *Management Science*, 53(2):308–322, 2007.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Murphy, S. A. A generalization error for Q-Learning. *J. Mach. Learn. Res.*, 6:1073–1097, July 2005.

Nichol, A., Pfau, V., Hesse, C., Klimov, O., and Schulman, J. Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*, 2018.

Precup, D., Sutton, R. S., and Singh, S. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning, 2000*, pp. 759–766. Morgan Kaufmann, 2000.

Quillen, D., Jang, E., Nachum, O., Finn, C., Ibarz, J., and Levine, S. Deep reinforcement learning for Vision-Based robotic grasping: A simulated comparative evaluation of Off-Policy methods. February 2018.

Raghu, M., Irpan, A., Andreas, J., Kleinberg, R., Le, Q., and Kleinberg, J. Can deep reinforcement learning solve erdos-selfridge-spencer games? In *International Conference on Machine Learning*, pp. 4235–4243, 2018.

Ross, S. and Bagnell, D. Efficient reductions for imitation learning. In *AISTATS*, pp. 661–668, 2010.

Theocharous, G., Thomas, P. S., and Ghavamzadeh, M. Personalized ad recommendation systems for life-time value optimization with guarantees. In *IJCAI*, pp. 1806–1812, 2015.

Thomas, P. and Brunskill, E. Data-Efficient Off-Policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 2139–2148, June 2016.

Thomas, P. S., Theocharous, G., and Ghavamzadeh, M. High-Confidence Off-Policy evaluation. *AAAI*, 2015.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

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

Zhang, A., Ballas, N., and Pineau, J. A dissection of overfitting and generalization in continuous reinforcement learning. June 2018a.

Zhang, C., Vinyals, O., Munos, R., and Bengio, S. A study on overfitting in deep reinforcement learning. April 2018b.

Zhang, S. and Sutton, R. S. A deeper look at experience replay. In *NeurIPS 2017 Deep RL Symposium*, 2017.

# Appendix for Off-Policy Evaluation of Generalization for Deep Q-Learning in Binary Reward Tasks

## A. Classification Error Bound

### A.1. Trajectory $\tau$ Return $1 \Leftrightarrow$ all $\mathbf{a}_t$ Good

For the forward direction, because $\tau$ ends in a success state, from any $\mathbf{s}_{t+1}$ the optimal policy $\pi^*$ could reach a success state, so all $(\mathbf{s}_t, \mathbf{a}_t)$ must be good.

For the reverse direction, if all $(\mathbf{s}_t, \mathbf{a}_t)$ are good, then $(\mathbf{s}_T, \mathbf{a}_T)$ must be good. Since $\mathbf{s}_{T+1}$ is a terminal state with no further actions, for $(\mathbf{s}_T, \mathbf{a}_T)$ to be good, we must have $r(\mathbf{s}_T, \mathbf{a}_T) = 1$.

### A.2. Proof of Theorem 1

By definition, $\pi$ succeeds if and only if at every $\mathbf{s}_t$, it selects a good $\mathbf{a}_t$. Since $\rho_{t,\mu,good}$ is defined as the state distribution conditioned on $\mathbf{a}_1, \cdots, \mathbf{a}_{t-1}$ being good, the failure rate can be written as

$$1 - R(\pi) = \sum_{t=1}^{T} Pr(\pi \text{ makes first mistake at time } t) \quad (1)$$

$$= \sum_{t=1}^{T} \epsilon_t \prod_{i=1}^{t-1} (1 - \epsilon_i) \quad (2)$$

$$\leq \sum_{t=1}^{T} \epsilon_t \leq T\epsilon \quad (3)$$

This gives $R(\pi) \geq 1 - T\epsilon$ as desired. The bound is tight when $\epsilon_1 = \epsilon_2 = \cdots = \epsilon_{T-1} = 0, \epsilon_T = T\epsilon$.

### A.3. Connection to Behavioral Cloning

Since every policy that only picks good actions achieves the optimal reward of 1, and $\epsilon$ is defined as the 0-1 loss over states conditioned on not selecting a bad action, we can view $\epsilon$ as the 0-1 behavior cloning loss to an expert policy $\pi^*$. Applying Theorem 2.1 of (Ross & Bagnell, 2010) gives a $O(T^2\epsilon)$ cost, compared to the $O(T\epsilon)$ cost derived above.

The difference in bound comes because (Ross & Bagnell, 2010) derive their proof in a general MDP whose cost is upper bounded by 1 at every timestep. If $\pi$ deviates from the expert, it receives cost 1 several times, once for every future timestep. In binary reward MDPs, we only receive this cost once, at the final timestep. Transforming the proof to incorporate our binary reward MDP assumptions lets us recover the $O(T\epsilon)$ upper bound from Appendix A. We briefly explain the updated proof, using notation from (Ross & Bagnell, 2010) to make the connection more explicit.

Define $\epsilon_t$ as the expected 0-1 loss at time $t$ for $\pi$ under the state distribution of $\pi^*$. Note this is the same as our definition of $\epsilon_t$. The MDP is defined by cost instead of reward: cost 0 for all timesteps except the final one, which is cost 0 or 1. Let $p_t$ be the probability $\pi$ hasn't made a mistake (w.r.t $\pi^*$) in the first $t$ steps, $d_t$ be the state distribution conditioned on no mistakes in the first $t-1$ steps, and $d'_t$ be the state distribution conditioned on $\pi$ making at least 1 mistake. In a general MDP with $0 \leq C(\mathbf{s}) \leq 1$, total cost $J(\pi)$ is bounded by $J(\pi) \leq \sum_{t=1}^{T} [p_{t-1}\mathbb{E}_{d_t}[C_\pi(\mathbf{s})] + (1 - p_{t-1})]$, where the 1st term is cost while following the expert and the 2nd term is a cost 1 if outside of the expert distribution. In a binary reward MDP, since $C(\mathbf{s}) = 0$ for all $t$ except $t = T$, we instead have

$$J(\pi) = p_{T-1}\mathbb{E}_{d_T}[C_\pi(\mathbf{s}_T)] + (1 - p_{T-1}) \quad (4)$$

Note $\mathbb{E}_{d_T}[C_\pi(\mathbf{s}_T)] = \epsilon_T$, and as shown in the original proof, $p_t \geq 1 - \sum_{i=1}^{t} \epsilon_i$. Using $p_{T-1}\mathbb{E}_{d_T}[C_\pi(\mathbf{s}_T)] \leq \mathbb{E}_{d_T}[C_\pi(\mathbf{s}_T)]$ recovers the $O(T\epsilon)$ bound, and again this is tight when $\epsilon_1 = \cdots = \epsilon_{T-1} = 0, \epsilon_T = T\epsilon$.

$$J(\pi) \leq \mathbb{E}_{d_T}[C_\pi(\mathbf{s}_T)] + \sum_{t=1}^{T-1} \epsilon_t = \sum_{t=1}^{T} \epsilon_T = T\epsilon \quad (5)$$

## B. Efficiently Computing the OPC Score

$$OPC(Q) = \pi_p \mathbb{E}_{pos}\left[1_{Q(s,a)>b}\right] - \mathbb{E}_{all}\left[1_{Q(s,a)>b}\right] \quad (6)$$

Suppose we have $N$ transitions, of which $N^+$ of them have positive labels. Imagine placing each $Q(\mathbf{s}, \mathbf{a})$ on the number line. Each $Q(\mathbf{s}, \mathbf{a})$ is annotated with a score, $-1/N$ for unlabeled transitions and $\pi_p/N^+ - 1/N$ for positive labeled transitions. We initialize a sweep line at $b < \min_{\mathcal{D}} Q(\mathbf{s}, \mathbf{a})$. Using this $b$ as the threshold gives OPC score $\pi_p - 1$. Sweep

this line across the number line until $b > \max_{\mathcal{D}} Q(\mathbf{s}, \mathbf{a})$. The OPC score only updates when this line passes a $Q(\mathbf{s}, \mathbf{a})$ in our dataset, and is updated based on the score annotated at $Q(\mathbf{s}, \mathbf{a})$, so this runs in $O(N)$ time. Given $\mathcal{D}$, we sort the $N$ transitions in $O(N \log N)$, annotate them appropriately, then compute the maximum over all OPC scores.

## C. Arguments for Choosing $\pi_p = 1$

Empirically, we found a positive class prior of $\pi_p = 1$ produces an OPE metric that correlates best with return. In this section, we speculate why this is the case.

Consider the practical computation of OPC presented in Appendix B. Suppose $\pi_b$, the policy collecting our validation set, succeeds with probability $p$. Then we have $N$ transitions, where $pN$ of them have positive labels. Each $Q(\mathbf{s}, \mathbf{a})$ is annotated with a score as described in Appendix B: $-1/N$ for unlabeled transitions and $\pi_p/(pN) - 1/N$ for positive labeled transitions.

The maximal OPC score will be the sum of all annotations within the interval $[b, \infty)$, for some $b$. For unlabeled transitions, the annotation is $-1/N$, which is negative. Suppose $\pi_p/(pN) - 1/N$ was negative as well. If every annotation is negative, then the optimal choice for $b$ is $b = \infty$, giving $OPC(Q) = 0$. This argument holds no matter what $Q(\mathbf{s}, \mathbf{a})$ we are evaluating, giving a score of 0 to every model, making the OPC score entirely independent of episode return. This degenerate case is clearly undesirable, and happens when $\pi_p/(pN) < 1/N$, or equivalently $\pi_p < p$.

To avoid this, $\pi_p$ must be larger than $p = R(\pi_b)$. If we are only allowed to pick a single prior $\pi_p$, that must generalize to arbitrary behavior policies $\pi_b$, then we should pick $\pi_p \geq R(\pi^*)$. In binary reward MDPs where $\pi^*$ can always succeed, this gives $\pi_p \geq R(\pi^*) = 1$, and since the prior is a probability that should satisfy $0 \leq \pi_p \leq 1$, choosing $\pi_p = 1$ is the only option.

To finish the argument, we must handle the case where we have a binary reward MDP where $R(\pi^*) < 1$. In a binary reward MDP, the only way to have $R(\pi^*) < 1$ is if the sampled initial state $\mathbf{s}_0$ is one where $(\mathbf{s}_0, \mathbf{a})$ is bad for all $\mathbf{a}$. From these $\mathbf{s}_0$, and all future $\mathbf{s}_t$ reachable from $\mathbf{s}_0$, the actions $\pi$ chooses do not matter - the final return will always be 0. It is reasonable to assume we only wish to compute the metric over states where our actions can impact reward, and within this part of the state space, we have $R(\pi^*) = 1$.

## D. Full Atari Results

Figure 1 is the same Atari figure from the main paper, except with a legend explaining each hyperparameter setting used. From top to bottom, the abbreviations mean:

- `DQN`: trained with DQN
- `DDQN`: trained with Double DQN
- `DQN_gamma9`: trained with DQN, $\gamma = 0.9$ (default is 0.99).
- `DQN2`: trained with DQN, using a different random seed
- `DDQN2`: trained with Double DQN, using a different random seed
- `DQN_lr1e4`: trained with DQN, learning rate $10^{-4}$ (default is $2.5 \times 10^{-4}$).
- `DQN_b64`: trained with DQN, batch size 64 (default is 32).
- `DQN_fixranddata`: The replay buffer is filled entirely by a random policy, then a DQN model is trained against that buffer, without pushing any new data.
- `DQN_fixranddata`: The replay buffer is filled entirely by a random policy, then a Double DQN model is trained against that buffer, without pushing any new data.

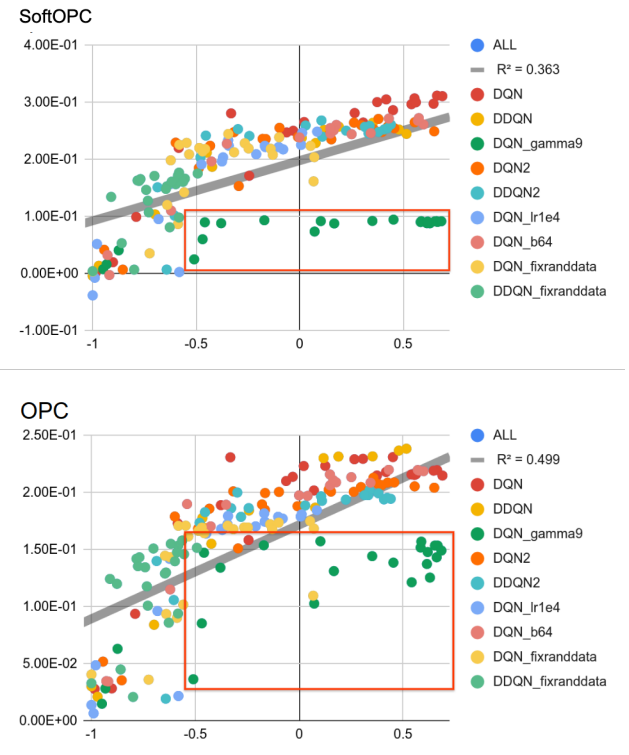


*Figure 1.* Scatterplot of episode return (x-axis) against metric (y-axis), for SoftOPC (top) and OPC (bottom), with legend explaining each color.

## E. SoftOPC Performance on Different Validation Datasets

Our validation dataset was collected from two policies, a poor policy with a success of 28%, and a better policy with a success of 51%. We divided the dataset based on the policy, then evaluated SoftOPC on both. Figure 2 shows the fit on these subsets of the validation dataset. The fit is slightly worse on the poor dataset, but the relationship between SoftOPC and episode reward is still clear.

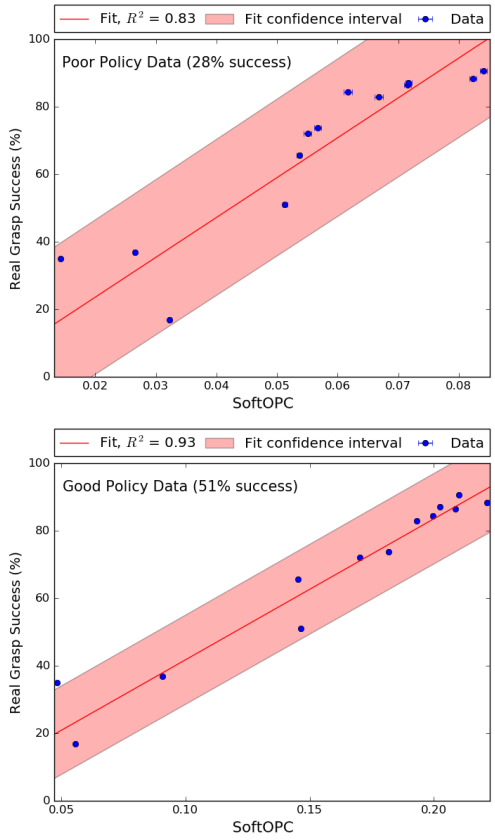


*Figure 2.* **SoftOPC versus the real grasp success for different data-collection behavior policies.** Top shows SoftOPC over a dataset from only the poor policy (28% success rate) and bottom shows a SoftOPC over data only from the better policy (51% success). A fitted regression line with its $R^2$ and confidence interval is also shown.

## F. Plots of Q-value Distributions

In Figure 3, we plot the Q-values of two real-world grasping models. The first is trained only in simulation and has poor real-world grasp success. The second is trained with a mix of simulated and real-world data. We plot a histogram of the average Q-value over each episode of validation set $\mathcal{D}$. The better model has a wider separation between successful Q-values and failed Q-values.

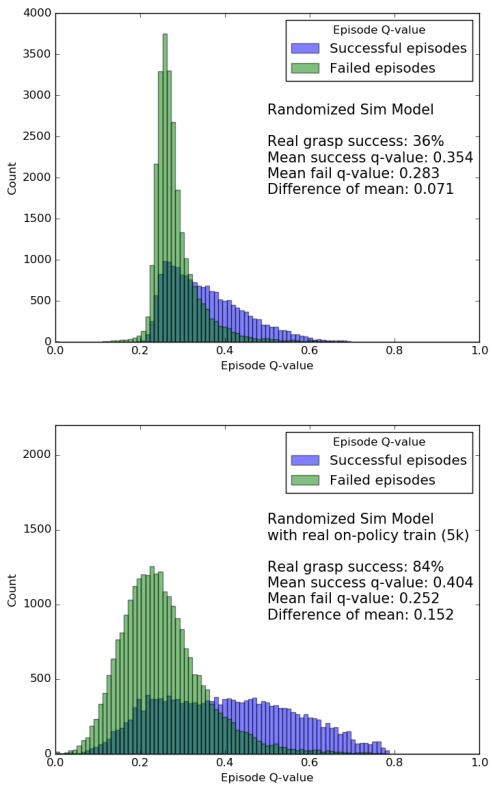


*Figure 3.* **Q-value distributions for successful and failed episodes.** (Top) Q-value distributions for successful and failed episodes in an off-policy data-set according to a learned policy with a poor grasp success rate of 36%. (Bottom) The same distributions after the learned policy is improved by fine-tuning with 5000 grasps on the real robots, achieving a 84% grasp success rate.

## G. SoftOPC or OPC?

We see SoftOPC and OPC correlate better with return than our baseline metrics, but comparison *between* SoftOPC and OPC is less clear. We suspect OPC performs better when the compared Q-functions have systematically different magnitudes. For example, models trained with a smaller $\gamma$ will have smaller Q-values over the entire state-action space. These systematic differences bias the SoftOPC to favor Q-functions with larger magnitudes.

In the tree environments, $Q(\mathbf{s}, \mathbf{a}) \in [0, +1]$ by construction. In the grasping environments, $Q(\mathbf{s}, \mathbf{a}) \in [0, 1]$, due to the network architecture ending in $\text{sigmoid}(x)$, bounding output to $[0, 1]$. In these experiments, SoftOPC did better. In Pong, $Q(\mathbf{s}, \mathbf{a})$ was not constrained, and OPC did better.

## References

Ross, S. and Bagnell, D. Efficient reductions for imitation learning. In *AISTATS*, pp. 661–668, 2010.