Quantile QT-Opt for Risk-Aware Vision-Based Robotic Grasping

Cristian Bodnar¹, Adrian Li², Karol Hausman³, Peter Pastor², Mrinal Kalakrishnan²

Abstract—The distributional perspective on reinforcement learning (RL) has given rise to a series of successful Q-learning algorithms, resulting in state-of-the-art performance in arcade game environments. However, it has not yet been analyzed how these findings from a discrete setting translate to complex practical applications characterized by noisy, high dimensional and continuous state-action spaces. In this work, we propose Quantile QT-Opt (Q2-Opt), a distributional variant of the recently introduced distributed Q-learning algorithm [11] for continuous domains, and examine its behaviour in a series of simulated and real vision-based robotic grasping tasks. The absence of an actor in Q2-Opt allows us to directly draw a parallel to the previous discrete experiments in the literature without the additional complexities induced by an actor-critic architecture. We demonstrate that Q2-Opt achieves a superior vision-based object grasping success rate, while also being more sample efficient. The distributional formulation also allows us to experiment with various risk-distortion metrics that give us an indication of how robots can concretely manage risk in practice using a Deep RL control policy. As an additional contribution, we perform experiments on offline datasets and compare them with the latest findings from discrete settings. Surprisingly, we find that there is a discrepancy between our results and the previous batch RL findings from the literature obtained on arcade game environments.

I. INTRODUCTION

The new distributional perspective on RL has produced a novel class of Deep Q-learning methods that learn a distribution over the state-action returns, instead of using the expectation given by the traditional value function. These methods, which obtained state-of-the-art results in the arcade game environments [4], [6], [5], present several attractive properties.

First, their ability to preserve the multi-modality of the action values naturally accounts for learning from a non-stationary policy, most often deployed in a highly stochastic environment. This ultimately results in a more stable training process and improved performance and sample efficiency. Second, they enable the use of risk-sensitive policies that no longer select actions based on the expected value, but take entire distributions into account. These policies can represent a continuum of risk management strategies ranging from risk-averse to risk-seeking by optimizing for a broader class of risk metrics.

Despite the improvements distributional Q-learning algorithms demonstrated in the discrete arcade environments, it is yet to be examined how these findings translate to practical, real-world applications. Intuitively, the advantageous properties of distributional Q-learning approaches should be particularly beneficial in a robotic setting. The value distributions can have a significant qualitative impact in robotic tasks, usually characterized by highly-stochastic and continuous state-action spaces. Additionally, performing safe control in the face of uncertainty is one of the biggest impediments to deploying robots in the real world, an impedance that RL methods have not yet tackled. In contrast, a distributional approach can allow robots to learn an RL policy that appropriately quantifies risks for the task of interest.

However, given the brittle nature of deep RL algorithms and their often counter-intuitive behaviour [8], it is not entirely clear if these intuitions would hold in practice. Therefore, we believe that an empirical analysis of distributional Q-learning algorithms in real robotic applications would shed light on their benefits and scalability, and provide essential insight for the robot learning community.

In this paper, we aim to address this need and perform a thorough analysis of distributional Q-learning algorithms in simulated and real vision-based robotic manipulation tasks. To this end, we propose a distributional enhancement of QT-Opt [11] subbed Quantile QT-Opt (Q2-Opt). The choice of QT-Opt, a recently introduced distributed Q-learning algorithm that operates on continuous action spaces, is dictated by its demonstrated applicability to large-scale vision-based robotic experiments. In addition, by being an actor-free generalization of Q-learning in continuous action spaces, QT-Opt enables a direct comparison to the previous results on the arcade environments without the additional complexities and compounding effects of an actor-critic-type architecture.

In particular, we introduce two versions of Q2-Opt, based on Quantile Regression DQN (QR-DQN) [6] and Implicit Quantile Networks (IQN) [5]. The two methods are evaluated on a vision-based grasping task in simulation and the real world. We show that these distributional algorithms achieve state-of-the-art grasping success rate in both settings, while also being more sample efficient. Furthermore, we experiment with a multitude of risk metrics, ranging from risk-seeking to risk-averse, and show that risk-averse policies can bring significant performance improvements. We also report on the interesting qualitative changes that different risk metrics induce in the robots’ grasping behaviour. As an additional contribution, we analyze our distributional methods in a batch RL scenario and compare our findings with an equivalent experiment from the arcade environments [1].
Deep learning has shown to be a useful tool for learning visuomotor policies that operate directly on raw images. Examples include various manipulation tasks, where related approaches use either supervised learning to predict the probability of a successful grasp [16], [13], [18] or learn a reinforcement-learning control policy [14], [20], [11].

Distributional Q-Learning algorithms have been so far a separate line of research, mainly evaluated on game environments. These algorithms replace the expected return of an action with a distribution over the returns and mainly vary by the way they parametrize this distribution. Bellemare et al. [4] express it as a categorical distribution over a fixed set of equidistant points. Their algorithm, CS1, minimizes the KL-divergence to the projected distributional Bellman target. A follow-up algorithm, QR-DQN [6], approximates the distribution by learning the outputs of the quantile function at a fixed set of points, the quantile midpoints. This latter approach has been extended by IQN [5], which reparametrized the critic network to take as input any probability \( \tau \) and learn the quantile function itself. Besides this extension, their paper also analyses various risk-sensitive policies that the distributional formulation enables. In this work, we apply these advancements to a challenging vision-based grasping task with a continuous action space.

Closest to our work is D4PG [3], a distributed and distributional version of DDPG [15] that achieves superior performance to the non-distributional version in a series of simulated continuous-control environments. In contrast to this work, we analyze a different variant of Q-learning with continuous action spaces, which allows us to focus on actor-free settings that are similar to the previous distributional Q-learning algorithms. Besides, we demonstrate our results on real robots on a challenging vision-based grasping task.

In this paper we consider a standard Markov Decision Process [19] formulation \( (S, A, r, p, \gamma) \), where \( s \in S \) and \( a \in A \) denote the state and action spaces, \( r(s, a) \) is a deterministic reward function, \( p(\cdot|s, a) \) is the transition function and \( \gamma \in (0, 1) \) is the discount factor.

As previously stated, we build our method on top of QT-Opt [11], a distributed Q-learning algorithm suitable for continuous action spaces. The algorithm trains a parameterized state-action value function \( Q_\theta(s, a) \) which is represented by a neural network with parameters \( \theta \). The cross-entropy method (CEM) [22] is used to iteratively optimize and select the best action for a given Q-function:

\[
\pi_\theta(s) = \arg \max_a Q_\theta(s, a)
\]

In order to train the Q-function, a separate process called the “Bellman Updater” samples transition tuples \( (s, a, r, s') \) containing the state \( s \), action \( a \), reward \( r \), and next state \( s' \) from a replay buffer and generates Bellman target values according to a clipped Double Q-learning rule [24], [23]:

\[
\hat{Q}(s, a, s') = r(s, a) + \gamma V(s')
\]

where \( V(s') = Q_\tilde{\theta}_1(s', \pi_\tilde{\theta}_2(s')) \), and \( \tilde{\theta}_1 \) and \( \tilde{\theta}_2 \) are the parameters of two delayed target networks. These target values are pushed to another replay buffer \( D \), and a separate training process optimizes the Q-function against a training objective:

\[
\mathcal{E}(\theta) = \mathbb{E}_{(s, a, s') \sim D} \left[ D(Q_\theta(s, a), \hat{Q}(s, a, s')) \right]
\]
IV. QUANTILE QT-OPT (Q2-OPT)

In Q2-Opt (Figure[1]), the value function no longer predicts a scalar value, but rather a vector \( q_\theta(s, a, \tau) \) that predicts the quantile function output for a vector of input probabilities \( \tau \), with \( \tau_i \in [0, 1] \) and \( i = 1, \ldots, N \). Thus the \( i \)-th element of \( q_\theta(s, a, \tau) \) approximates \( F_{s,a}(\tau_i) \), where \( F_{s,a} \) is the CDF of the value distribution belonging to the state action pair \( (s, a) \). In practice, this means that the neural network has a multi-headed output. However, unlike QT-Opt where CEM optimizes directly over the Q-values, in Quantile QT-Opt, CEM maximizes a scoring function \( \psi: \mathbb{R}^N \rightarrow \mathbb{R} \) that maps the vector \( q \) to a score \( \psi(q) \):

\[
\pi_\theta(s, \tau) = \arg \max_a \psi(q_\theta(s, a, \tau))
\]  
(4)

Similarly, the target values produced by the “Bellman updater” are vectorized using a generalization of the clipped Double-Q-learning rule from QT-Opt:

\[
q_\theta(s, a, s', \tau', \tau'') = r(s, a) 1 + \gamma \psi(s', \tau', \tau'') \\
\psi(s', \tau', \tau'') = q_\theta(s', \tau_\theta(s', \tau'', \tau'))
\]  
(5)

where \( 1 \) is a vector of ones, and, as before, \( \theta_1 \) and \( \theta_2 \) are the parameters of two delayed target networks. Even though this update rule has not been considered so far in the distributional RL literature, we find it effective in reducing the overestimation in the predictions.

In the following sections, we present two versions of Q2-Opt based on two recently introduced distributional algorithms: QR-DQN and IQN. The main differences between them arise from the inputs \( \tau, \tau', \) and \( \tau'' \) that are used. To avoid overloading our notation, from now on we omit the parameter subscript in \( q_\theta, \hat{q}_\theta \) and replace it with an index into these vectors \( q_i, \hat{q}_i \).

A. Quantile Regression QT-Opt (Q2R-Opt)

In Quantile Regression QT-Opt (Q2R-Opt), the vectors \( \tau, \tau', \tau'' \) in \( q \) and \( \hat{q} \) are fixed. They all contain \( N \) quantile midpoints of the value distribution. Concretely, \( q_i(s, a, \tau) \) is assigned the fixed quantile target \( \tau_i = \frac{\tau_{i-1} + \tau_i}{2} \) with \( \tau_0 = 0 \). The scoring function \( \psi(\cdot) \) takes the mean of this vector, reducing the \( N \) quantile midpoints to the expected value of the distribution when \( N \) is sufficiently large. Because \( \tau, \tau', \tau'' \) are always fixed we consider them implicit and omit adding them as an argument to \( q \) and \( \hat{q} \) for Q2R-Opt.

The quantile heads are optimized by minimizing the Huber [9] quantile regression loss:

\[
\rho^\kappa_{\tau}(\delta_{ij}) = |\tau - \mathbb{I}[\delta_{ij} < 0]| \frac{\mathcal{L}_{\kappa}(\delta_{ij})}{\kappa}
\]

\[
\mathcal{L}_{\kappa}(\delta_{ij}) = \begin{cases} 
\frac{1}{2} \delta_{ij}^2, & \text{if } |\delta_{ij}| \leq \kappa \\
\kappa(1 - |\delta_{ij}| - \frac{1}{2} \kappa), & \text{otherwise}
\end{cases}
\]  
(6)

for all the pairwise TD-errors:

\[
\delta_{ij} = q_j(s, a, s') - q_i(s, a)
\]  
(7)

Thus, the network is trained to minimize the loss function:

\[
\mathcal{E}(\theta) = \mathbb{E}_{(s,a,s') \sim D} \left[ \sum_{i=1}^{N} \mathbb{E}_j \left[ \rho^\kappa_{\tau}(\delta_{ij}) \right] \right]
\]  
(8)

B. Quantile Function QT-Opt (Q2F-Opt)

In Q2F-Opt, the neural network itself approximates the quantile function of the value distribution, and therefore it can predict the inverse CDF for any \( \tau \). Since \( \tau, \tau', \tau'' \) are no longer fixed, we explicitly include them in the arguments of \( q \) and \( \hat{q} \). Thus, the TD-errors \( \delta_{ij} \) take the form:

\[
\delta_{ij} = \hat{q}_j(s, a, s', \tau', \tau'') - q_i(s, a, \tau),
\]  
(9)

where \( \tau_i \sim U[0, 1], \tau'_j \sim U[0, 1] \) and \( \tau''_j \sim U[0, 1] \) are sampled from independent uniform distributions. Using different input probability vectors also decreases the correlation between the networks. Note that now the length of the prediction and target vectors are determined by the lengths of \( \tau \) and \( \tau' \). The model is optimized using the same loss function as the one from Equation [8]

C. Risk-Sensitive Policies

The additional information provided by a value distribution compared to the (scalar) expected return gives birth to a broader class of policies that go beyond optimizing for the expected value of the actions. Concretely, the expectation can be replaced with any risk metric, that is any function that maps the random return to a scalar quantifying the risk. In Q2-Opt, this role is played by the function \( \psi \) that acts as a risk-metric. Thus the agent can handle the intrinsic uncertainty of the task in different ways depending on the specific form of \( \psi \). It is important to specify that this uncertainty is generated by the environment dynamics \( p \) and the (non-stationary) policy collecting the real robot rollouts and that it is not a parametric uncertainty.

We distinguish two methods to construct risk-sensitive policies for Q2R-Opt and Q2F-Opt, each specific to one of the methods. In Q2R-Opt, risk-averse and risk-seeking policies can be obtained by changing the function \( \psi(\cdot) \) when selecting actions. Rather than computing the mean of the target quantiles, \( \psi(\cdot) \) can be defined as a weighted average over the quantiles \( \psi(q(s, a)) = \frac{1}{N} \sum_i w_i q_i(s, a) \). This sum produces a policy that is in between a worst-case and best-case action selector and, for most purposes, it would be preferable in practice over the two extremes. For instance, a robot that would consider only the worst-case scenario would most likely terminate immediately since this strategy, even though it is not useful, does not incur any penalty. Behaviours like this have been encountered in our evaluation of very conservative policies.

In contrast, Q2F-Opt provides a more elegant way of learning risk-sensitive control policies by using risk-distortion metrics [25]. Recently, Majmidar and Pavone [17] have argued for the use of risk distortion metrics in robotics. They proposed a set of six axioms that any risk measure should meet to produce reasonable behaviour and showed that risk distortion metrics satisfy all of them. However, to the best of our knowledge, they have not been tried on real robotic applications.

The key idea is to use a policy:

\[
\pi_\theta(s, \beta(\tau)) = \arg\max_a \psi(q(s, a, \beta(\tau))),
\]
where $\beta : [0, 1] \rightarrow [0, 1]$ is an element-wise function that distorts the uniform distribution that $\tau$ is effectively sampled from, and $\psi(\cdot)$ computes the mean of the vector as usual. Functions $\beta$ that are concave induce risk-averse policies, while convex function induce risk-seeking policies.

In our experiments, we consider the same risk distortion metrics used by Dabney et al. [5]: the cumulative probability weighting (CPW) [7], the standard normal CDF-based metric proposed by Wang [26], the conditional value at risk (CVaR) [21], Norm [5], and a power law formula (Pow) [5]. Concretely, we plug in different parameters for these distortions. CPW(0.71) is known to be a good model human behaviour [27], Wang(−.75), Pow(−2), CVaR(0.25) and CVaR(0.4) are risk-averse. Norm(3) decreases the weight of the distribution’s tails by averaging 3 uniformly sampled $\tau$. Ultimately, Wang(.75) produces risk-seeking behaviour.

Due to the relationships between Q2F-Opt and the literature of risk distortion measures, we focus our risk-sensitivity experiments on the metrics mentioned above and leave the possibility of trying different functions $\psi(\cdot)$ in Q2R-Opt for future work.

D. Model Architecture

To maintain our comparisons with QT-Opt, we use very similar architectures for Q2R-Opt and Q2F-Opt. For Q2R-Opt, we modify the output layer of the standard QT-Opt architecture to be a vector of size $N$, rather than a scalar. For Q2F-Opt, we take a similar approach to [5], and we embed every $\tau_i$ using a series of $n = 64$ cosine basis functions:

$$\phi_j(\tau_i) := \text{ReLU}\left(\sum_{i=0}^{n-1} \cos(\pi i \tau_i) w_{ij} + b_j\right)$$

We then perform the Hadamard product between this embedding and the convolutional features. Another difference in Q2F-Opt is that we replace batch normalization [10] in the final fully-connected layers with layer normalization [2]. We notice that this better keeps the sampled values in the range allowed by our MDP formulation. The three architectures are all included in a single diagram in Figure 2.

V. EXPERIMENTAL SETUP

We consider the problem of vision-based robotic grasping for our evaluations. In our grasping setup, the robot arm is placed at a fixed distance from a bin containing a variety of objects and tasked with grasping any object. The MDP specifying our robotic manipulation task provides a simple binary reward to the agent at the end of the episode: 0 for a failed grasp, and 1 for a successful grasp. To encourage the robot to grasp objects as fast as possible, we use a time step penalty of $-0.01$ and a discount factor $\gamma = 0.9$. The state is represented by a $472 \times 472$ RGB image; the actions are a mixture of continuous 4-DOF tool displacements in $x, y, z$ with azimuthal rotation $\phi$, and discrete actions to open and close the gripper, as well as to terminate the episode.

In simulation, we grasp from a bin containing 8 to 12 randomly generated procedural objects (Figure 2). For the first 5,000 global training steps, we use a procedural exploration policy. The scripted policy is lowering the end effector at a random position at the level of the bin and attempts to grasp. After 5,000 steps, we switch to an $\epsilon$-greedy policy with $\epsilon = 0.2$.

In the real world, we train our model from an offline dataset. For evaluation, we attempt 6 consecutive grasps from a bin containing 6 objects without replacement, repeated across 5 rounds. We perform this experiment in parallel on 7 robots, resulting in a total of 210 grasp attempts. All the robots use a similar object setup consisting of two plastic bottles, one metal can, one paper bowl, one paper cup, and one paper cup sleeve. In the results section, we report the success rate over the 210 attempts.

VI. RESULTS

In this section, we present our results on simulated and real environments. In simulation, we perform both online and offline experiments, while for the real world, the training is exclusively offline.

A. Simulated Environment

Figure 3 shows the mean success rate as a function of the global training step together with the standard deviation across five runs for QT-Opt, Q2R-Opt and Q2F-Opt. Because Q2-Opt and QT-Opt are distributed systems, the global step does not directly match the number of environment episodes used by the models during training. Therefore, to understand the sample efficiency of the algorithm, we also include in Figure 4 the success rate as a function of the total number of environment episodes added to the buffer.

We find the results to be aligned with the findings on the arcade-game environments. Q2F-Opt achieves the best grasp success rate out of all the considered methods, stabilizing around 92.8%, while also being significantly more sample efficient than QT-Opt. Q2R-Opt exhibits an intermediary performance. Despite being less sample efficient than Q2F-Opt, it still learns significantly faster than QT-Opt in the late stages of training. The final performance statistics are given in full detail in Table I.

B. Risk-Sensitive Policies

We evaluate a series of risk distortion measures with different parameter settings in simulation. Figure 5 shows the success rate for various measures used in Q2F-Opt. We notice that risk-averse policies (Wang(−.75), Pow(−2), CVaR) are generally more stable in the late stages of training and achieve a higher success rate. Pow(−2) remarkably achieves 95% grasp success rate. However, being too conservative can also be problematic. Particularly, the CVaR(0.25) policy

| Model                | Mean Success | Success Std | Median Success |
|----------------------|--------------|-------------|----------------|
| QT-Opt               | 0.903        | 0.005       | 0.903          |
| Q2R-Opt (Ours)       | 0.923        | 0.006       | 0.924          |
| Q2F-Opt (Ours)       | **0.928**    | **0.001**   | **0.928**      |

TABLE I: Final sim success rate statistics. The proposed distributional methods achieve higher success rate.
C. Real-World Environment

For the real-world evaluation, we train our models offline from a 76GB dataset of real-world experiences, containing 500,000 episodes, collected over many months using a multitude of control policies. The chaotic physical interactions specific to real-world environments and the diversity of policies used to gather the experiences make this experiment an ideal scenario for applying distributional RL. Furthermore, this experiment is also of practical importance for robotics since any increase in grasp success rate from offline data reduces the amount of costly online training that has to be performed to obtain a good policy.

We report in Table III the grasp success rate statistics for all the considered models. We find that the best risk-averse version of Q2-Opt achieves an impressive 17.6% higher success rate than QT-Opt. While the real evaluation closely matches the model hierarchy observed in sim, the success rate differences between the models are much more significant.

The risk-averse distortion measures of Q2F-Opt have a significant qualitative impact. They show a tendency to reposition the gripper in positions that are more favourable or to move objects around to make grasping easier. CVaR(0.4), the most conservative metric we tested in the real world, presented a particularly interesting behaviour of intentionally dropping poorly grasped objects to attempt a better regrasp. The CVaR policy mainly used this technique when
Fig. 5: Average success rate over five runs for different risk-sensitive policies in sim. Most risk averse policies perform better, but extremely conservative ones like CVaR(.25) can become unstable. The risk-seeking policy Wang(.75) performs worst than the others.

Table II: Final sim success rate statistics for different risk distortion metrics. Risk-averse policies have the best success rate, with Pow(−2) at the top.

| Model          | Success | Std  | Median |
|----------------|---------|------|--------|
| Q2F-Opt CVAR(0.25) | 0.933   | 0.013| 0.941  |
| Q2F-Opt CVAR(0.4)  | 0.938   | 0.008| 0.938  |
| Q2F-Opt CPW(0.71)  | 0.928   | 0.003| 0.925  |
| Q2F-Opt Wang(0.75) | 0.898   | 0.012| 0.893  |
| Q2F-Opt Wang(-0.75)| 0.942   | 0.007| 0.944  |
| Q2F-Opt Pow(-2.0) | **0.950** | **0.004**| **0.952** |

Fig. 6: Sim off policy success rate for data collected during a full training run. None of the methods can achieve the final performance of the policy trained online (90%).

VII. CONCLUSION

In this work, we have examined the impact that value distributions have on practical robotic tasks. Our proposed method, Q2-Opt, achieved state-of-the-art success rates on simulated and real vision-based robotic grasping tasks, while also being significantly more sample efficient than its non-distributional equivalent, QT-Opt. The success rate improvements brought by Q2-Opt by training only from real offline data could drastically reduce the number of environment interactions required to train a good robot control policy. Additionally, we have shown how safe reinforcement learning control can be achieved through risk-sensitive policies and reported the range of interesting behaviours these policies produce in practice. As a final contribution, we evaluated the proposed distributional methods in a batch RL setting similar to that of Agarwal et al. [1] and showed that, unfortunately, their findings do not translate to the continuous grasping environment presented in this work.
ACKNOWLEDGMENT

We would like to give special thanks to Ivonne Fajardo and Noah Brown for overseeing the robot operations.

APPENDIX

A. Real World Setup

Figure 7 illustrates our workspace setup with the objects used for evaluation. We considered a diverse set of objects containing both rigid and soft bodies, presenting different textures, colours and shapes.

B. Risk-sensitive behaviours

An interesting metric we considered to analyze the behaviour of different risk-sensitive policies is the number of broken gripper fingers throughout the entire evaluation process. Figure 8 plots these numbers. Even though we do not have a statistically significant number of samples, we believe this data provides minimal insight into the behaviour of these policies.

C. Offline Evaluation

Besides the dataset obtained from the training of a model, we are also analyzing our distributional methods on two more fixed datasets containing transitions:

- Collected by a scripted stochastic exploration policy with 46% success rate.
- Produced by a policy with 89% grasp success rate.

Figure 9 plots the results for the first scenario, while Figure 10 for the second. Perhaps surprisingly, the models achieve a higher success rate on the scripted exploration dataset than on the replay buffer dataset. Figure 10 plots the results for the first scenario, while Figure 10 for the second. Perhaps surprisingly, the models achieve a higher success rate on the scripted exploration dataset than on the replay buffer dataset. Figure 10 plots the results for the first scenario, while Figure 10 for the second. Perhaps surprisingly, the models achieve a higher success rate on the scripted exploration dataset than on the replay buffer dataset. Figure 10 plots the results for the first scenario, while Figure 10 for the second. Perhaps surprisingly, the models achieve a higher success rate on the scripted exploration dataset than on the replay buffer dataset.
REFERENCES

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. Striving for simplicity in off-policy deep reinforcement learning. ArXiv, abs/1907.04543, 2019.
[2] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. ArXiv, abs/1607.06450, 2016.
[3] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. ArXiv preprint arXiv:1804.08617, 2018.
[4] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In International Conference on Machine Learning, pages 449–458, 2017.
[5] Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. Implicit quantile networks for distributional reinforcement learning. In International Conference on Machine Learning, pages 1104–1113, 2018.
[6] Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[7] Rodrigo González and George Wu. On the shape of the probability weighting function. Cognitive Psychology, 38:129–166, 1999.
[8] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. ArXiv, abs/1709.06560, 2017.
[9] Peter J. Huber. Robust estimation of a location parameter. Ann. Math. Statist., 35(1):73–101, 03 1964.
[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, ICML’15, pages 448–456. JMLR.org, 2015.
[11] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In Conference on Robot Learning, pages 651–673, 2018.
[12] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. ArXiv, abs/1906.00949, 2019.
[13] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. The Journal of Machine Learning Research, 17(1):1334–1373, 2016.
[14] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International Journal of Robotics Research, 37(4-5):421–436, 2018.
[15] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. CoRR, abs/1509.02971, 2015.
[16] Jeffrey Mahler, Jacky Liang, Sheridil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. 2017.
[17] Anirudha Majumdar and Marco Pavone. How should a robot assess risk? Towards an axiomatic theory of risk in robotics. ArXiv, abs/1710.11040, 2017.
[18] Leontina Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 3406–3413, 2015.
[19] Martin L. Puterman. Markov decision processes: Discrete stochastic dynamic programming. In Wiley Series in Probability and Statistics, 1994.
[20] Deirdre Quillen, Eric Jang, Ofir Nachum, Chelsea Finn, Julian Ibarz, and Sergey Levine. Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 6284–6291. IEEE, 2018.
[21] R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. Journal of Risk, 2:21–41, 2000.
[22] Reuven Y. Rubinstein and Dirk P. Kroese. The cross-entropy method. In Information Science and Statistics, 2004.
[23] Richard S. Sutton and Andrew G. Barto. Introduction to Reinforcement Learning. MIT Press, Cambridge, MA, USA, 1st edition, 1998.
[24] Hado van Hasselt. Double q-learning. In NIPS, 2010.
[25] Shaun Wang. Premium calculation by transforming the layer premium density. ASTIN Bulletin, 26(1):7192, 1996.
[26] Shaun S. Wang. A class of distortion operators for pricing financial and insurance risks. Journal of Risk and Insurance, pages 15–36, 2000.
[27] George Wu and Richard Gonzalez. Curvature of the probability weighting function. Management Science, 42(12):1676–1690, 1996.