Avoiding Value Estimation Error in Off-Policy Deep Reinforcement Learning

Jared Markowitz **Jesse Silverberg** **Gary Collins**
Johns Hopkins University Applied Physics Laboratory
{jared.markowitz,jesse.silverberg,gary.collins}@jhuapl.edu

Abstract

The most popular approaches for off-policy deep reinforcement learning (DRL) with continuous action spaces include policy improvement steps where a learned state-action value (Q) function is maximized over selected batches of data. These algorithms also use the Q-function in the target of its own update, and end up overestimating the Q-values as a result. To combat this overestimation, these algorithms take a minimum over multiple Q estimates in the value update target. We examine a setting, common in real-world applications, where improperly balancing these opposing sources of bias may have disastrous consequences: stochastic environments with reward functions comprised of multiple negatively-correlated terms. Reward terms corresponding to conflicting objectives will be negatively correlated; in expectation, gains in one will be accompanied by losses in the other. We find that standard approaches consistently fail to approach optimal performance when applied to a suite of robotic tasks in this category. We trace the failure to erroneous Q estimation and propose a novel off-policy actor-critic algorithm that remediates the problem through the use of a policy gradient. Our algorithm significantly outperforms baseline approaches across such tasks, drastically reducing the total cost incurred by the agent throughout training.

1 Introduction

Motivated by real-world applications, we here consider off-policy learning in systems that have continuous action spaces, stochastic dynamics, and force the agent to meaningfully consider both incentives and costs. Application areas where this setting is relevant include robotics (e.g., navigation in the presence of obstacles, robotic surgery), resource allocation under uncertainty (e.g., when both performance and efficiency must be considered), and financial decision-making (e.g., where the risk of losses exists in the pursuit of gains).

We find that current state-of-the-art off-policy approaches for continuous action spaces may struggle when applied to such problems. We trace the failure to opposing sources of bias in the learned state-action value (Q) function, derive an algorithm that largely avoids such bias, and show our approach to be a broadly robust method for DRL.

2 Preliminaries

We consider the standard MDP reinforcement learning setting (Sutton & Barto, 1998). For off-policy learning, we can write the RL objective for a policy π with parameters θ as

$$J(\theta) \approx \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\big[V^{\pi_\theta}(\mathbf{s})\big] = \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})}\big[Q^{\pi_\theta}(\mathbf{s},\mathbf{a})\big]. \tag{1}$$

One strategy for maximizing $J(\theta)$ is to compute a policy gradient for Eq. (1) (Degris et al., 2012):

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\substack{\mathbf{s}\sim\mathcal{D}\\ \mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})}}\Big[\nabla_\theta \log\big(\pi_\theta(\mathbf{a}|\mathbf{s})\big)Q^{\pi_\theta}(\mathbf{s},\mathbf{a}) + \nabla_\theta Q^{\pi_\theta}(\mathbf{s},\mathbf{a})\Big] \approx \mathbb{E}_{\substack{\mathbf{s}\sim\mathcal{D}\\ \mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})}}\Big[\nabla_\theta \log\big(\pi_\theta(\mathbf{a}|\mathbf{s})\big)A^{\pi_\theta}(\mathbf{s},\mathbf{a})\Big], \tag{2}$$

where $A^{\pi_\theta}(\mathbf{s}, \mathbf{a}) = Q^{\pi_\theta}(\mathbf{s}, \mathbf{a}) - V^{\pi_\theta}(\mathbf{s})$ is the advantage function. There are two commonly-used alternatives to Eq. (2). The first is the deterministic policy gradient (Silver et al., 2014, DPG), which considers a deterministic π_θ and is used by TD3:

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\Big[\nabla_\theta \pi_\theta(\mathbf{s}) \nabla_{\mathbf{a}} Q^{\pi_\theta}(\mathbf{s}, \mathbf{a})\big|_{\mathbf{a}=\pi_\theta(\mathbf{s})}\Big]. \tag{3}$$

The second is the reparameterization trick (used by SAC), which results in a lower variance gradient estimate than Eq. (2) (Kingma et al., 2015):

$$\nabla_\theta J(\theta) = \mathbb{E}_{\mathbf{s}\sim\mathcal{D}, \xi\sim\mathcal{N}(\mathbf{0}, I)}\Big[\nabla_\theta \min_{i=1,2} Q^{\pi_\theta}_{\phi_i}\big(\mathbf{s}, \mathbf{a}_\theta(\mathbf{s}, \xi)\big) - \alpha \nabla_\theta \log\big(\pi_\theta(\mathbf{a}_\theta(\mathbf{s}, \xi) \mid \mathbf{s})\big)\Big]. \tag{4}$$

where the policy is implicitly parameterized by a neural network $\mathbf{a}_\theta(\mathbf{s}, \xi) = \tanh\big(\mu_\theta(\mathbf{s}) + \sigma_\theta(\mathbf{s}) \odot \xi\big)$.

3 Addressing Biases in Off-Policy Deep Reinforcement Learning

In this work, we consider Safety Gym, a suite of robotic navigation tasks with obstacles originally proposed for studying safe RL (Ray et al., 2019). These environments separately yield *rewards* (incentives), for making progress toward the goal, and *costs*, for colliding with obstacles. This is a frequently-used testbed for constrained RL methods, which consider the task of maximizing rewards subject to constraint conditions on the accumulated costs (Altman, 1999). Alternatively, we can modify these environments to work with standard (unconstrained) RL methods by constructing a reward function $r(\mathbf{s}, \mathbf{a}) = i(\mathbf{s}, \mathbf{a}) - \beta c(\mathbf{s}, \mathbf{a})$, with incentives $i(\mathbf{s}, \mathbf{a})$, nonnegative costs $c(\mathbf{s}, \mathbf{a})$, and fixed scalar weight β.

We find this seemingly straightforward setting to be a catastrophic failure mode for common off-policy algorithms, characterized by **significant Q-value underestimation** and **high temporal difference (TD) error** (Fig. 1).

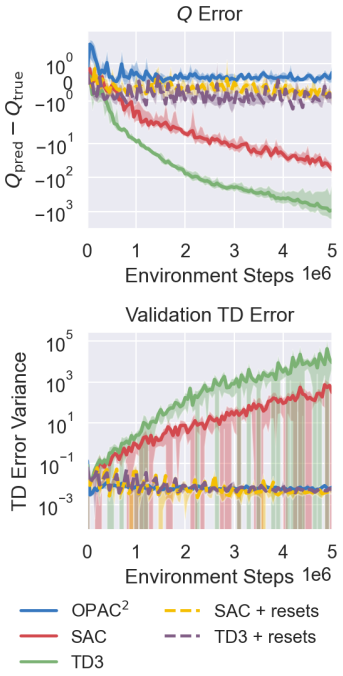

Figure 1: Learning diagnostics on PointGoal2. SAC and TD3 suffer from significant value underestimation error. Q_{true} is estimated using discounted Monte Carlo returns.

3.1 Why Do Current Approaches Fail?

Most state-of-the-art off-policy methods both regress the Q-function toward itself in its own update and train a policy to maximize Q. Combining these elements induces Q *overestimation* bias (Thrun & Schwartz, 1993; Hasselt, 2010; Fujimoto et al., 2018). Positive estimation error induces a preference in the policy update toward actions whose Q-values are overestimated. This may prevent the discovery of optimal actions and leads to preferential selection of actions with erroneously large Q-values in the target of the value update.

To address this bias, modern off-policy approaches employ Clipped Double Q-learning (Fujimoto et al., 2018). That is, they train two Q-functions with initial parameters ϕ_1, ϕ_2 and use the minimum of their respective target networks (delayed; parameters $\bar{\phi}_1, \bar{\phi}_2$) in the target of the value updates. Other algorithms go further by taking the minimum over an ensemble of Q-functions in the value update (Lan et al., 2020; Chen et al., 2021). In all cases, the idea is to introduce Q *underestimation* bias to combat the accumulation of overestimation errors (Fujimoto et al., 2018).

As noted in Fujimoto et al. (2018), Q-value estimates may be written as a combination of rewards provided by the environment, and error terms. That is, $Q^\pi_\phi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi\big[\sum_{i=t}^{\infty} \gamma^{i-t}(r_i - \delta_i) \mid \mathbf{s}, \mathbf{a}\big]$, where $\delta(\mathbf{s}, \mathbf{a}) \equiv r + \gamma Q^\pi_\phi(\mathbf{s}', \mathbf{a}') - Q^\pi_\phi(\mathbf{s}, \mathbf{a})$ is the temporal difference (TD) error. Variance in Q-value estimates may therefore be attributed to either variance in future rewards assigned by the

environment or function approximation error. *Aleatoric uncertainty*—for instance, from stochasticity in the environment or partial observability—contributes through variance of future rewards. Regardless of the source, increased variance in Q-value estimates leads to more significant underestimation bias from Clipped Double Q-learning, since the expected minimum of a set of random variables decreases as the variance of the random variables increases (Fujimoto et al., 2018).

The authors of Chen et al. (2021) show that as long as the amount of Q estimation error is bounded, the minimum of a sufficiently large number of Q estimates will be an *underestimate*. With only one Q network, only overestimation bias is present and learning is generally poor (Fig. 4). If Q-values are problematically underestimated when taking the minimum of two Q networks, minimizing over additional Q estimates will only exacerbate the problem.

3.2 An Algorithm that (Mostly) Avoids Bias

To address the potentially harmful effects of opposing Q estimation bias highlighted above, we revisit the policy gradient update (Eq. 2) proposed by Degris et al. (2012). We first observe that this update is not as prone to overestimation bias as those based on the deterministic policy gradient (Eq. 3) or reparameterization trick (Eq. 4). This, in turn, allows us to omit Clipped Double Q-learning and avoid its associated underestimation bias entirely.

Additionally, we may configure the updates to the value networks in our approach to mitigate estimation bias. Similar to Haarnoja et al. (2018) (but without Clipped Double Q-learning), we use

$$\mathcal{E}_Q(B) = \sum_i \Big[Q(\mathbf{s}_i, \mathbf{a}_i) - \big(r_i + \gamma(1-d_i) V_{\bar{\psi}}(\mathbf{s}'_i)\big) \Big]^2$$

$$\mathcal{E}_V(B) = \sum_i \Big[V(\mathbf{s}_i) - Q(\mathbf{s}_i, \mathbf{a}_{i,\pi}) \Big]^2$$

as the losses to minimize in the Q and V updates, with a target network for V providing stability. Here $\mathbf{a}_{i,\pi}$ indicates an action sampled from $\pi(\cdot|\mathbf{s}_i)$; the subscript i runs over a batch sampled from the replay buffer. Note that the policy update (Eq. 2) will reinforce *positive* error on Q but *negative* error on V. In using Q and V in each other's targets, we provide a counterbalance for these potential sources of error. Additionally, the use of V in the learning target for Q provides smoothness, which may be beneficial. Finally, we employ tanh activations in our neural networks, removing a potential source of poor numerical stability. Empirically, we found this strategy to result in only very slight Q *overestimation* in every environment tested. This did not adversely impact policy learning; hence Clipped Double Q-learning was not required. We refer to our approach as **O**ff-**P**olicy **A**ctor-**C**ritic, **SQUA**shed and **RE**gularize**D** (**OPAC**^{2}), and provide pseudocode in Appendix A.

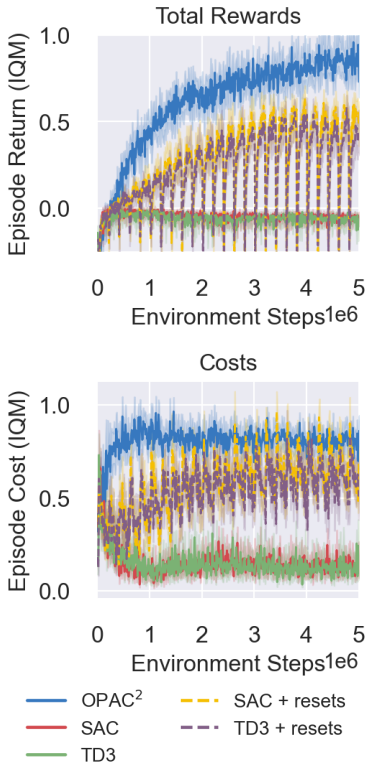

Figure 2: Our approach, OPAC^2, collects the highest levels of total reward (considering incentives and costs; top panel), outperforming all baselines. SAC and TD3 completely fail to learn.

4 Experiments

We evaluated all robots and tasks for the most obstacle-rich (level 2) Safety Gym environments. Full details are given in Appendix B. SAC and TD3 largely fail to learn competent policies in this set of environments (Fig. 2 and Appendix F). In addition to measuring the error in the learned Q estimates, we also report the variance of the validation TD error, which is computed as the square of the TD error δ of the learned Q-function over held-out batches of experience not seen in training (Li et al., 2023). In every environment, SAC and TD3 have greater validation TD error variance, greater

Q estimation error, and converge to policies with lower cost but far lower accumulation of incentives. Critically, SAC and TD3 significantly *underestimate* Q-values in most of these environments. By considering only the costs incurred by the agents (Fig. 2, bottom), we can determine that SAC and TD3 are incurring very little cost while also receiving little incentives (i.e., they have learned to do nothing). This may be a result of Clipped Double Q-learning providing targets that are, on average, too low. Since reward enters the learning algorithm only through this target, the end result is overly conservative behavior.

4.1 Partial Remediation through Resetting

Periodic network resets have recently been studied as a strategy for reducing Q estimation error (Nikishin et al., 2022; Li et al., 2023). We found this addition to be hugely beneficial for SAC and TD3 in some of the environments where they fail. Resetting allows SAC and TD3 to consistently achieve smaller Q estimation error than OPAC^2, but not as strong overall performance (Fig. 2). Typically, SAC and TD3 with resetting still act more conservatively than OPAC^2 (Appendix F).

4.2 Algorithm Analysis

We performed a systematic ablation of the algorithmic components of OPAC^2 in order to elucidate their impact on Safety Gym performance. We additionally included the original version of SAC (Haarnoja et al., 2018), augmented with optimized entropy regularization (as introduced in Haarnoja et al. (2019)). We find the inclusion of *both* a value network and a policy gradient to be critical in this setting; including only one is insufficient (Fig. 3). Interestingly, we found that we were able to remove the target value network from OPAC^2 entirely without significantly compromising its learning stability. We also found that OPAC^2 did not benefit from resetting, likely because its minimal value estimation errors did not require correction (Appendix D).

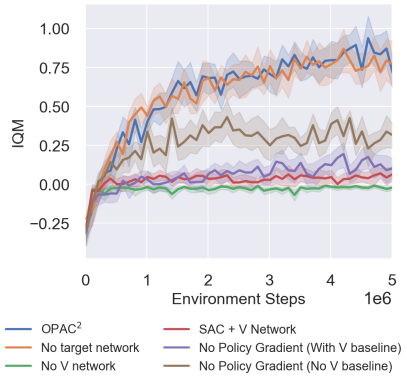

Figure 3: Ablation of OPAC^2

5 Related Work

Our method builds on the off-policy actor-critic introduced in Degris et al. (2012), applied in differing forms over the past decade (Wang et al., 2017; Espeholt et al., 2018; Gu et al., 2017), and summarized in Levine et al. (2020). We compare it with more popular methods for off-policy DRL with continuous action spaces (Haarnoja et al., 2019; Fujimoto et al., 2018). The resetting strategy that we employ was explored by Nikishin et al. (2022) and subsequently further evaluated and extended (D'Oro et al., 2023; Schwarzer et al., 2023). Its ability to curtail TD error, as well as the role of TD error in limiting off-policy DRL more generally, was discussed in Li et al. (2023).

6 Conclusion

In this work, we identify a failure mode of popular off-policy DRL methods for continuous action spaces. In contrast to the typical problem of value overestimation, we present environments that induce underestimation errors in current algorithms. These findings illustrate the utility of exploring different problem settings when designing algorithms; the conditions responsible for the failures we observe are not present in more popular simulation benchmarks (Tunyasuvunakool et al., 2020).

Motivated by real-world applications, we provide a novel algorithm that is much less prone to value estimation error, enabling learning on these environments. Our algorithm uses the same number of networks (and fewer parameters, owing to V requiring only a state as input) while providing strong learning in environments with high aleatoric uncertainty and conflicting objectives.

Broader Impact Statement

While our work is immediately targeted at making safer robotic agents, it is true that others could repurpose our work for malicious applications. We encourage the authors of any subsequent work to consider the societal impacts of future results.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC press, 1999.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=AY8zfZm0tDd.

Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. In *International Conference on Machine Learning*, 2012.

Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OpC-9aBBVJe.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJ3rcZcxl.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2019.

Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020. doi: 10.1038/s41586-020-2649-2.

Hado Hasselt. Double q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (eds.), *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper_files/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. doi: 10.1109/MCSE.2007.55.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015.

Diederik P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick, 2015.

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Bkg0u3Etwr.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

Qiyang Li, Aviral Kumar, Ilya Kostrikov, and Sergey Levine. Efficient deep reinforcement learning requires regulating overfitting. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=14-kr46GvP-.

Jared Markowitz, Ryan W. Gardner, Ashley Llorens, Raman Arora, and I-Jeng Wang. A risk-sensitive approach to policy optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(12):15019–15027, Jun. 2023. doi: 10.1609/aaai.v37i12.26753. URL https://ojs.aaai.org/index.php/AAAI/article/view/26753.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16828–16847. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/nikishin22a.html.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. 2019.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv:1707.06347, 2017.

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level Atari with human-level efficiency. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 30365–30380. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/schwarzer23a.html.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 387–395. JMLR.org, 2014. URL http://dblp.uni-trier.de/db/conf/icml/icml2014.html#SilverLHDWR14.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

Sebastian Thrun and A. Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of 4th Connectionist Models Summer School*. Erlbaum Associates, June 1993.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

Guido Van Rossum and Fred L Drake Jr. *Python tutorial*, volume 620. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2. URL https://doi.org/10.1038/s41592-019-0686-2.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HyM25Mqel.

A Pseudocode

Algorithm 1 Off-Policy Actor-Critic, SQUAshed and REgularizeD (OPAC^2)

Input: Initial policy parameters θ; Q parameters ϕ; V parameters ψ, entropy weight α
Initialize V target network parameters: $\bar{\psi} \leftarrow \psi$
for iteration $k \in [0, \ldots, K-1]$ **do**
  for step $s \in [0, \ldots, S-1]$ **do** ▷ Typically take just one step ($S = 1$)
    Sample $\mathbf{a} \sim \pi_{\theta_k}(\cdot|\mathbf{s})$; observe $\mathbf{s}' \sim p(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$ ▷ One step in MDP
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}, \mathbf{a}, \mathbf{s}', r(\mathbf{s}, \mathbf{a}))\}$ ▷ Update replay buffer
  end for
  for gradient step $g \in [0, \ldots, G-1]$ **do**
    Sample batch $B = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i, d_i)\}$ from $\mathcal{D}$
    Update Q: $\phi \leftarrow \phi - \lambda_\phi \nabla_\phi \mathcal{E}_Q(B)$
    Sample $\mathbf{a}_{i,\pi} \sim \pi(\mathbf{a}|\mathbf{s}_i)$
    Update V: $\psi \leftarrow \psi - \lambda_\psi \nabla_\psi \mathcal{E}_V(B)$
    Compute $A(\mathbf{s}_i, \mathbf{a}_{i,\pi}) = Q(\mathbf{s}_i, \mathbf{a}_{i,\pi}) - V(\mathbf{s}_i)$ ▷ Normalize
    Sample $\mathbf{a}_{\mathrm{rp},\pi} \sim \pi(\cdot|\mathbf{s}_i)$ with reparameterization trick
    Compute π loss: $\mathcal{E}_\pi(B) = \sum_i [\alpha \log \pi_\theta(\mathbf{a}_{\mathrm{rp},\pi}|\mathbf{s}_i) - A(\mathbf{s}_i, \mathbf{a}_{i,\pi}) \log(\pi_\theta(\mathbf{a}_{i,\pi}|\mathbf{s}_i)]$
    Update π: $\theta \leftarrow \theta - \lambda_\theta \nabla_\theta \mathcal{E}_\pi(B)$
    Compute α loss: $\mathcal{E}_\alpha(B) = -\alpha [\log(\pi(\mathbf{a}_{i,\pi}|\mathbf{s}_i)) + \mathcal{H}_{\mathrm{target}}]$
    Update α : $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha \mathcal{E}_\alpha(B)$
    Update target V network parameters: $\bar{\psi} \leftarrow \rho \bar{\psi} + (1-\rho)\psi$
  end for
end for

B Experimental Details

Hyperparameters are listed in Table 1. For experiments in Safety Gym (which are described more fully below), we used a learning rate of 10^{-4} for all "Car" and "Point" experiments (matching the setting in the OpenAI `safety-starter-agents` accompanying the suite). Because we observed instability in all algorithms when using the "Doggo" robot on the "Button" task, we employed a learning rate of 5×10^{-5} for all experiments with Doggo. All methods shared the same learning rate for the logarithm of the entropy weight α (5×10^{-4}).

Throughout, all neural networks considered were multilayer perceptrons, with two hidden layers of 256 units each. As is standard, ReLU activations were used for SAC and TD3. We chose tanh activations for OPAC^2 in order to match on-policy methods with a similar policy update. We evaluated tanh activations on SAC and TD3 as well, but found them to typically perform slightly worse than the ReLU activations. For experiments involving resetting, we followed the practice of Nikishin et al. (2022) of resetting all networks and optimizers (except for those corresponding to the learned temperature) every 200k environment steps.

The policy networks output the mean values of a multivariate normal distribution with diagonal covariance. For OPAC^2, control variances were optimized. Variances were independent of state for all experiments for Car and Point robots in the Goal and Push tasks. They varied with state for all experiments with the Doggo robot, as well as for the Car and Point robots on the Button task. For SAC, control variances always varied with state.

As mentioned in Section 4 of the main text, we chose to evaluate our approach using the OpenAI Safety Gym (Ray et al., 2019). This choice was governed by our desire to test in conditions with clear cost-incentive trade-offs, significant stochasticity, and adequate complexity. Additionally, many popular algorithms have already been evaluated on Safety Gym.

PARAMETER	VALUE
Discount	0.99
Replay Buffer Size	10^6
Optimizer	Adam (Kingma & Ba, 2015)
Network Hidden Layers	2
Network Hidden Units (per layer)	256
Batch Size	256
Target Network Update Interval	1
τ (target network averaging)	0.995
Initial Exploration (steps)	10000

Table 1: Shared hyperparameters for SAC, TD3, and OPAC[2]

The environments chosen were the most obstacle-rich of the publicly available environments. We considered all combinations of the 3 robots and 3 tasks, for a total of 9 different environment configurations. The Point robot is constrained to the 2D plane and has two control dimensions: one for moving forward/backward and one for turning. The Car robot also has two control dimensions, corresponding to independently actuated parallel wheels. It has a freely rotating wheel and, while it is not constrained to the 2D plane, typically remains in it. The Doggo robot is a quadrupedal robot with bilateral symmetry and 12 control dimensions. Several types of obstacles and tasks were present in the environments we evaluated. In all cases, the robot is given a fixed amount of time (1000 steps) to complete the prescribed task as many times as possible and is motivated by both sparse and dense reward contributions. In the "Goal" environments, the robot must navigate to a series of randomly-assigned goal positions, with a new target being assigned as soon as a goal is reached. In the "Button" environments, the robot must reach and press a sequence of goal buttons while avoiding other buttons. In the "Push" task, the robot must push a box to a series of goal positions. The set of obstacles are different for each task; among the three environments there are a total of five different constraint elements (hazards, vases, incorrect buttons, pillars, and gremlins), each with different dynamics. See Ray et al. (2019) for further details.

All of our experiments used a single indicator for overall cost at each time step (the Safety Gym default). Each cost event (i.e., the robot contacting an obstacle) was assigned a fixed (negative) weight in the reward function. The Car and Point configurations used penalty weights that matched Markowitz et al. (2023), and the penalty weights for Doggo configurations are listed in Table 2.

We report all results over 5 random seeds. Per-environment learning curves are reported with shaded regions representing $\pm$ standard deviation. For aggregating performance across tasks, we report the interquartile mean (IQM) with 95% bootstrapped confidence intervals, as described in Agarwal et al. (2021). In order to aggregate results on Safety Gym, we scale the rewards from tasks relative to the final *incentive* level achieved by an agent trained on these environments *without considering costs*. That is, if we write rewards r as the difference of incentives i and costs $c \geq 0$ scaled by constant β, $r = i - \beta c$, we compare the reward r achieved by our agents for $\beta > 0$ with the incentives i achieved by agents trained with $\beta = 0$. For IQM plots of costs, we scale relative to the cost levels incurred by those same agents. The specific agents used for comparison were the better of PPO (Schulman et al., 2017) or TRPO (Schulman et al., 2015) on each environment after 10M steps, as reported by Ray et al. (2019). For rewards, this was chosen to serve as a loose *upper* bound on the possible performance of an agent trained with costs. For costs, the choice was made for consistency.

B.1 Implementation Details

Our code is written for Python 3.9 (Van Rossum & Drake Jr, 1995, PSF). The environments we use depend on the OpenAI Safety Gym (Ray et al., 2019, MIT) and MuJoCo version 2.1.0 (Todorov et al., 2012, Apache 2.0). Other major dependencies are PyTorch version 2.3.0 (Paszke et al., 2019,

ENVIRONMENT	PENALTY
DoggoGoal2	0.025
DoggoButton2	0.0125
DoggoPush2	0.00625

Table 2: Penalty weights for Doggo environments. The "2" indicates these were the highest difficulty environments.

BSD-3), `numpy` version 1.23.5 (Harris et al., 2020, BSD-3), `scipy` version 1.13.10 (Virtanen et al., 2020, BSD-3), and `matplotlib` version 3.3.4 (Hunter, 2007, BSD-3).

Safety Gym experiments (with 5 seeds each) can be run to 5M steps on CPU (AMD EPYC 9654; internal cluster) in 40–60 hours. Since we tested other configurations of our algorithm during development that are not included in the paper, it is difficult to estimate the total amount of compute used for the project. A rough estimate is 1500 CPU core-days.

C Overestimation and Underestimation Biases in SAC and TD3

With only 1 Q function, SAC and TD3 tend to drastically *overestimate* Q-values (Fig. 4, right). With Clipped Double Q-learning this tendency is reduced, and may even result in *underestimation*.

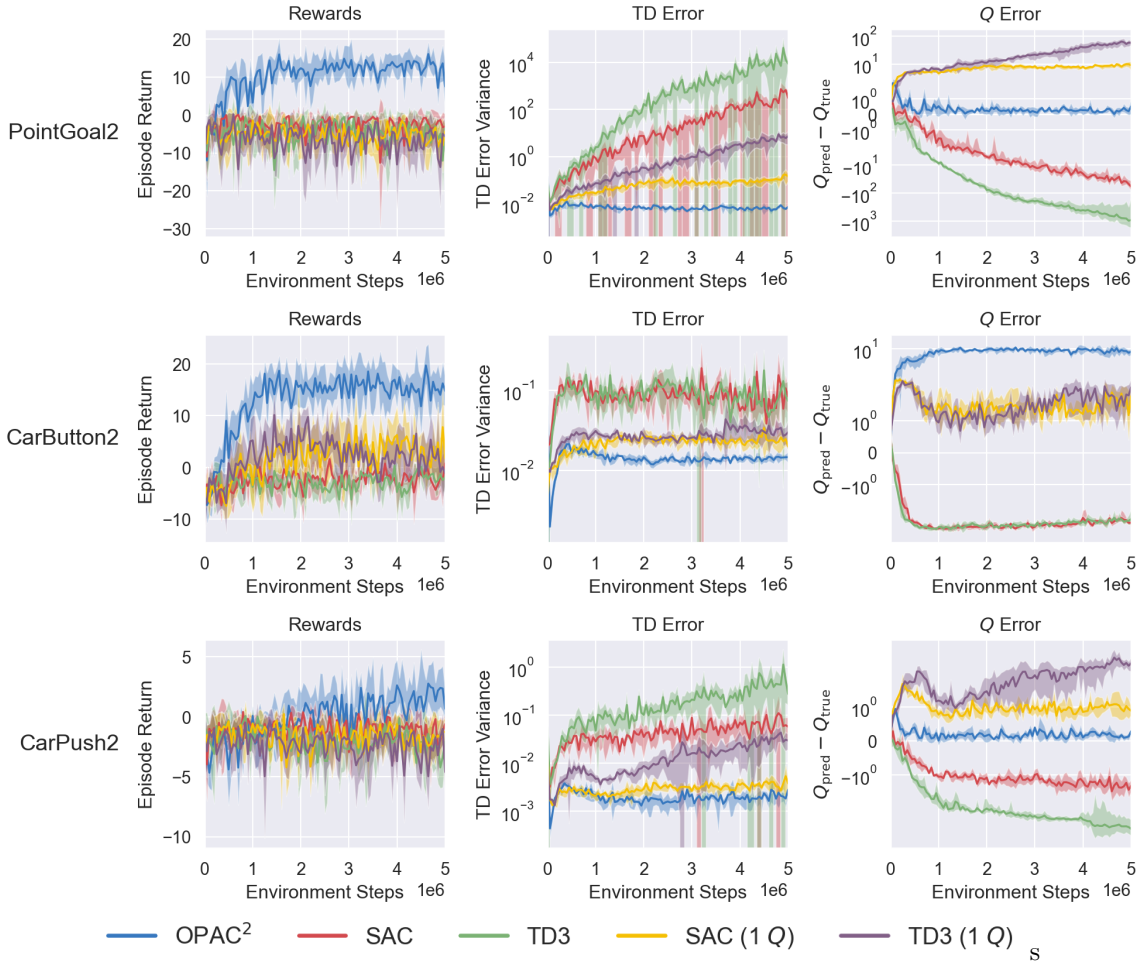

Figure 4: Reward and diagnostics showing the drastic effect of Clipped Double Q-learning in SAC and TD3 on TD error variance and Q error.

D Resetting OPAC^2

Given the performance gains we observed from augmenting SAC and TD3 with resetting, we considered its application in conjunction with OPAC^2. Unsurprisingly, we found that resetting actually had a negative effect on the performance of OPAC^2 (Figure 5). Resetting is intended to correct value estimation error, which is less of a concern with OPAC^2 than SAC or TD3. Without this benefit, the primary impact of resetting on OPAC^2 is a periodic disruption in training.

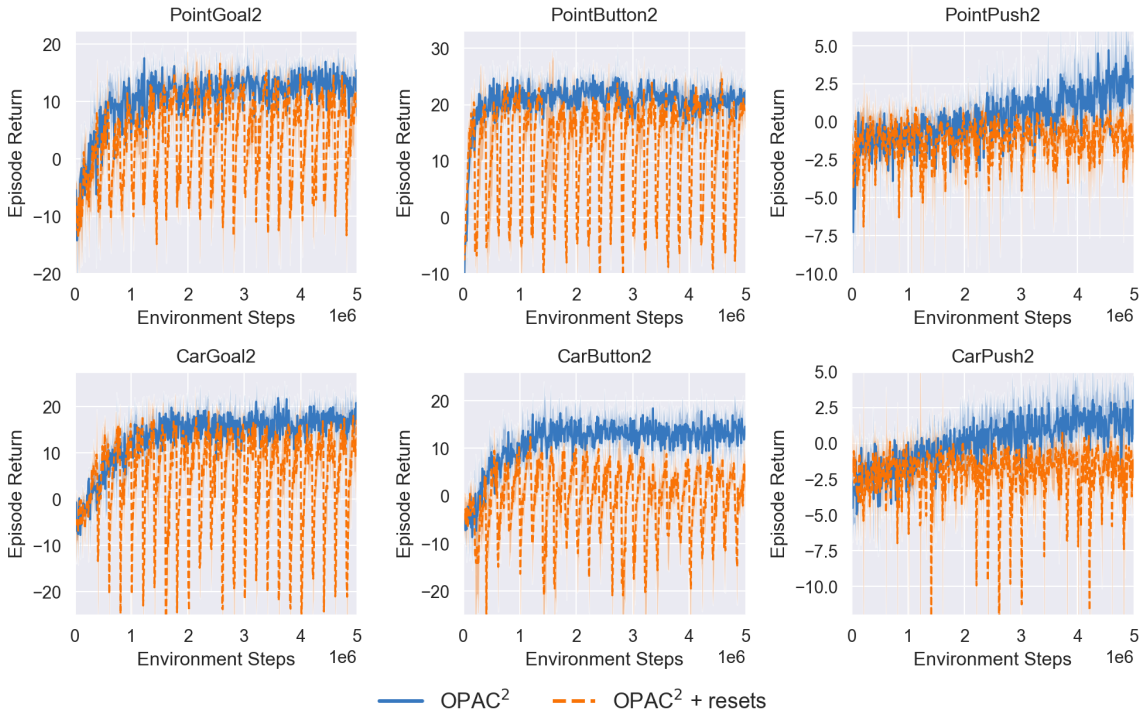

Figure 5: Periodically resetting OPAC^2 results in worse performance across environments.

E Entropy Regularization Strategies

Below we illustrate the effect of different entropy regularization strategies paired with OPAC^2 on two Doggo Safety Gym environments. We find that an entropy bonus slightly outperforms the "maximum-entropy" approach of SAC (or no entropy regularization at all) on these environments.

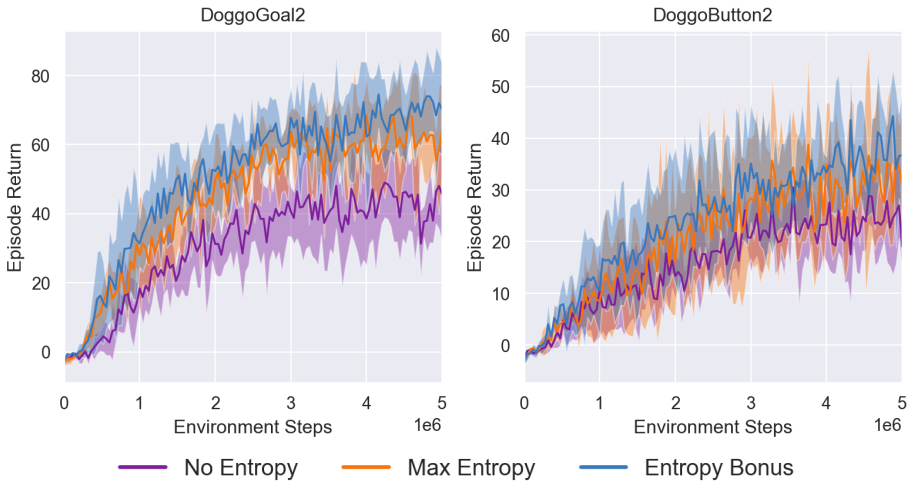

Figure 6: Evaluation of different entropy regularization strategies on Doggo Safety Gym environments.

F Full Safety Gym Results

In Fig. 7, we present per-environment reward curves for our experiments on Safety Gym. Note that these traces incorporate both incentives and costs (i.e., we plot $r = i - \beta c$). We observe that SAC and TD3 struggle to learn at all. Their performance improves when resetting is added, but not to the level of OPAC^2. In Fig. 8, we show the costs accumulated by the agents in each environment. Notice that SAC and TD3 always accumulate less cost than OPAC^2. Because the optimization goal is *rewards*, this amounts to overly conservative behavior by SAC and TD3. The mechanism underlying this is described in Section 3. Note that resetting raises the cost levels of SAC and TD3 somewhat, enabling them to collect more incentives.

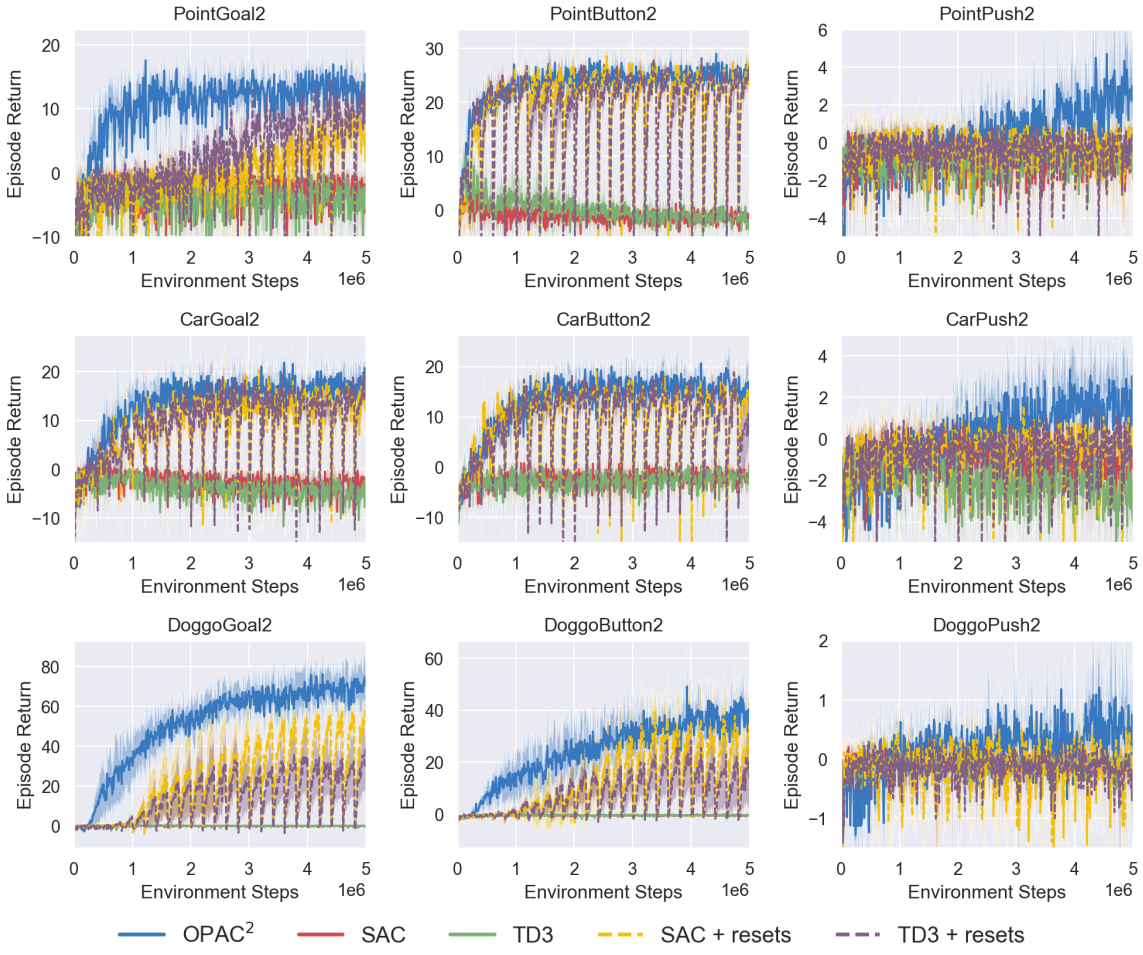

Figure 7: Per-environment learning curves for all nine Level 2 Safety Gym navigation environments.

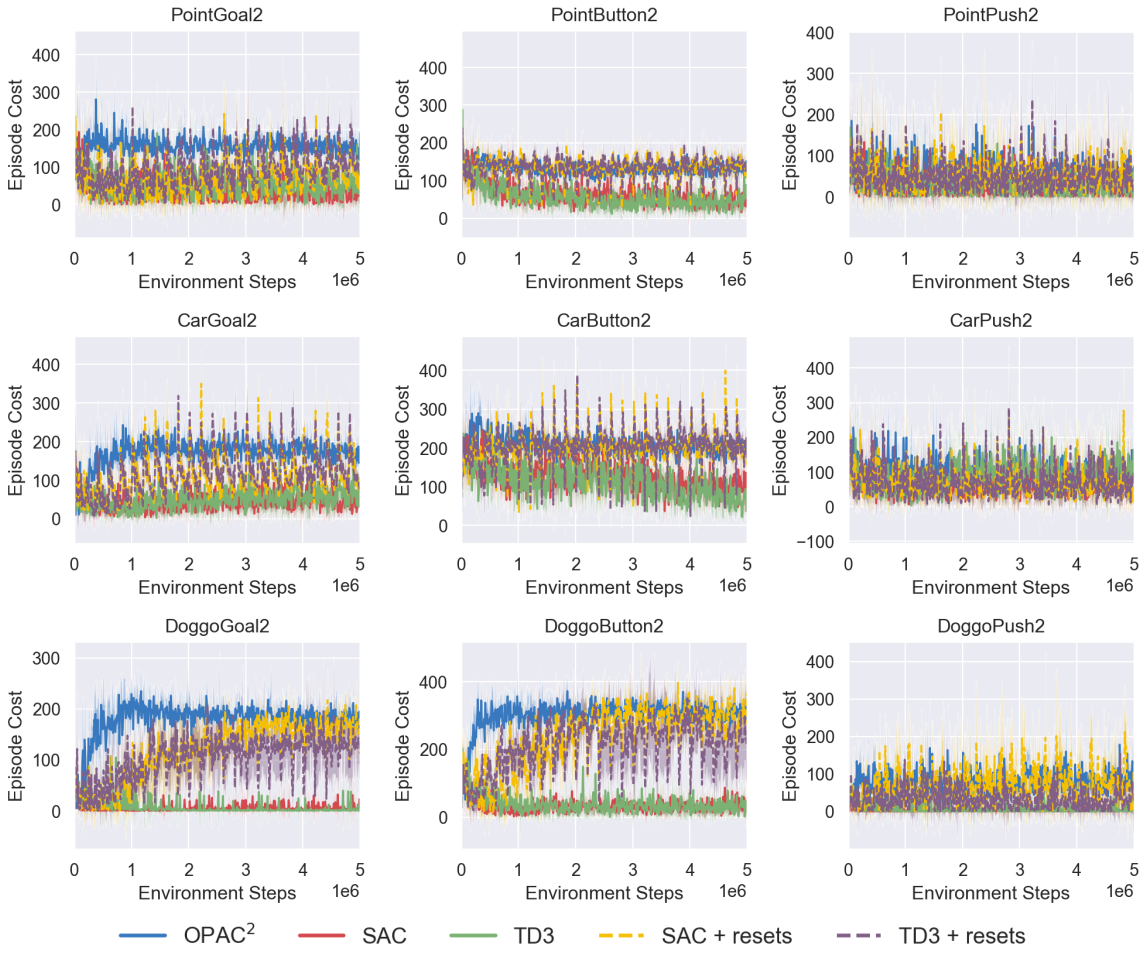

Figure 8: Per-environment learning curves depicting costs incurred on all nine Level 2 Safety Gym navigation environments.