Fearful Goal Generation for Reliable Policy Learning

Charlie Gauthier∗, Florian Golemo†, Glen Berseth∗, Liam Paull∗‡
Mila - Quebec AI Institute
* Université de Montréal, † McGill University, ‡ CIFAR AI chair
{charlie.gauthier,golemofl,glen.berseth,paulll}@mila.quebec

Abstract: By learning from experience, reinforcement learning (RL) methods learn from their environments adaptively, making them a promising direction for generalizable robots. However, training robotic goal-conditioned RL policies often requires careful tuning of reward functions, especially because of early termination problems: giving the RL agent negative feedback (such as about crashes) can cause it to be overly cautious. And yet, we desire agents that know to avoid such crashes as they can damage robot hardware. We propose DEIMOS, a novel safety-aware automatic training goal selector that requires no safety constraint Jacobian or conditional value at risk computation, nor any difference in observation space or reward shaping, and no extra neural parameters at deployment, making it ideal for agents acting on complex robotic morphologies. We showcase the efficacy of our method on a challenging quadruped locomotion and manipulation task. We empirically show that using our method, policies are tuned to optimize for safety, producing populations of final agents that crash less often than populations trained with baseline curricula. Their reward performance is also similarly improved.

Keywords: Quadruped robots, Curriculum learning, Reinforcement learning

1 Introduction

In deep reinforcement learning (DRL), policies iteratively adapt as they explore their environments, making it one of the most promising avenues when it comes to control policies for generalizable robots. While DRL is promising, it is often prohibitively costly to train in the real world. A solution is training in simulation, which often requires fine-tuning a noisy final policy on the hardware, which can then result in damaging the robot. To take full advantage of the generality of learning from experience, we require methods that allow us to be confident that the learned policy will perform well and act *safely* on the final hardware with less adaptation.

Solely through goal selection during training, we seek to produce policies that are safer at evaluation time while still accomplishing their goals to the best of their ability. We propose a novel automatic goal selection method that is simple and easy to implement. In this strategy, we introduce a parameterized goal selector to the policy's training, the "director". In parallel to the policy's training, we train a failure predictor that learns to predict if the policy will crash on a given goal. By querying the failure predictor, the director can retrieve goals that are adequately hard and better shape the policy's training. Our strategy requires no explicit constraints, reward shaping, input space augmentation, computing auxiliary safety violation optimization terms, or additional policy neural parameters, all of which can make training more brittle [1][2]. This makes it ideal for robotic learning. We demonstrate our strategy on a challenging quadruped locomotion and manipulation task. Empirically, our strategy greatly reduces trained policy population spread and mean for total evaluation crashes. It similarly improves the final agents' ability to perform the task (quantified through mean reward).

2 Background

We consider learning over a goal-conditioned Markov Decision Process (GMDP) that is defined by a tuple $< \mathcal{S}, \mathcal{G}, \mathcal{A}, P, R, \rho_0, \rho_g, \mathcal{T}, \gamma >$ where $\mathcal{S}$ is the robotic state and sensor information, $\mathcal{G}$ is a continuous vector space describing goals, P is the transition dynamics, R is the reward function, ρ_0 is the initial state distribution, and γ is a reward discount factor. A subset $\mathcal{S}_c$ of $\mathcal{S}$ are undesirable terminal states which we call "crash-states". As RL for robotics is negatively impacted by early termination problems [3][4], we consider crash-states with no associated negative reward term. As for ρ_g and $\mathcal{T}$, they respectively describe the goal distribution over $\mathcal{G}$ and the amount of timesteps between goal changes. During both training and deployment, a new goal $g \sim \rho_g$ is sampled every $[\mathcal{T} > 1]$ GMDP timesteps. We call these goal-change events "interactions" and they are described by tuples $< s_t, g_t, g_{t+1} >$, with t the current GMDP timestep, s_t and g_t the current state and goal, and g_{t+1} the goal at timestep $t + 1$. During deployment, to indiscriminately cover every goal in every state, ρ_g is a uniform distribution over $\mathcal{G}$.

Also important for this work are contextual bandits, which can be seen as single-step MDPs. As such, the tuple defining them is much simpler: $< S, A, R >$, where S is the context given to the decision policy, A is the set of actions available to the policy, and $R : S \times A \to \mathbb{R}$ is the reward function. A subclass of bandits of particular interest are Bernoulli bandits, where the reward is a binary success indicator $R : S \times A \to \mathbb{1}\{\text{success}\}$. While seemingly restrictive, Bernoulli bandits are actually very common in game theory and hold important relevance for this work. A common heuristic for solving Bernoulli bandits with discrete action spaces is Thompson Sampling, wherein a state-action value function is represented by a Beta distribution parameterized by the historical success/failure of each action during training [5]. When making a decision, the agent samples an arbitrary number of possible actions, rates them by the value function, and selects the highest-valued. With adequate tuning, such a strategy is close to optimal [6].

3 Method

We posit that an optimal distribution of training goals exists such that π_θ crashes less at evaluation time then when using a uniform goal distribution. An oracle goal selector would select goals that are neither too easy (already solvable) nor too hard (intractable), yet still suitably handle catastrophic forgetting (by occasionally sampling easier goals) and exploration (by sampling harder goals). The mechanics of learner agents being incredibly complex, we pose two major assumptions to render this problem tractable: (1) The probability of π_θ crashing upon receiving a given goal is a useful signal for goal selection, and (2) The oracle would presumably select goals with intermediary crash probabilities often, but also select easy and hard goals occasionally. Named after the policy's own fear of crashing, we call the crash probability the "fear score", given by a "fear function". We introduce a new agent to the policy's training, the "director" ρ_g^d, who uses a fear function to gain traction over training goal selection. Because π_θ is non-stationary, the likelihood of a given goal causing a crash is itself non-stationary, and thus the fear function must be learned online, alongside π_θ. This naturally implies a game between the policy and the director. This formulation follows the intuition of numerous other works on adversarial training and goal selection [7][8][9]. The main difference in our work is that we eschew the traditional three-agent setup of generator, discriminator, and agent; instead we rely on only agent and director. Additionally, we target both deployment safety and task performance, not just performance. Finally, we explicitly limit the director's power such that it is not fully adversarial by tuning its selection strategy over the fear scores. Otherwise, it is too good at crashing π_θ, and produces useless policies (see Appendix 7.1).

For π_θ, the game at hand is about selecting an optimal sequence of actions in a GMDP to survive goals set during an interaction with the director while also acting upon them optimally. For ρ_g^d, the game is about selecting a single goal g_{t+1} every $\mathcal{T}$ GMDP timestep that will cause the policy to crash, limited by the suboptimal sampling strategy we impose, conditioned on the information that the policy is in given state-goal pair $< s_t, g_t >$. Both agents optimize an expectation over a simple binary win condition $W_\pi^d(s_t, g_t, g_{t+1}) = \mathbb{1}\{\pi(\cdot|g_{t+1}) \text{ crashes within } \mathcal{T} \text{ steps}\}$. The policy implicitly minimizes W^d_π by default as part of its RL objective (early termination in infinite-horizon problems implies a lower cumulative sum of rewards). The director seeks to maximize W^d_π.

We notice that the description of the director's task corresponds to a Bernoulli bandit defined by $< S = \mathcal{S} \times \mathcal{G}, A = \mathcal{G}, R = W^d_\pi >_\pi$. The policy's non-stationary nature also influences the bandit's possible contexts (the policy visits different states depending on its training) and optimal actions and reward structure (the goals that crash π_θ change over training). Because of the bandit's continuous action space and non-stationary nature, standard Thompson Sampling is ill-fitted. Following our intuition, we gain traction over the non-stationarity by learning a fear function $\mathcal{F}_\Psi : \mathcal{S} \times \mathcal{G} \times \mathcal{G} \to [0, 1]$ on-policy. This neural network takes in interactions and outputs scores representing crash likelihoods. For the director, $\mathcal{F}_\Psi$ directly corresponds to a value function over W^d_π. We effectively obtain ρ^d_g by "inverting" $\mathcal{F}_\Psi$: after sampling many possible g_{t+1} goals and concatenating them to a single $< s_t, g_t >$, we score the goals using the fear network. By taking the *argmax*, we recover an approximation of the optimal adversarial goal (subject to the limitations we apply to ρ^d_g). Because our novel goal selection strategy uses the fear network at its core, we call it **D**read **E**nforced **I**nteractons for **M**ore **O**ptimal **S**ampling (DEIMOS).

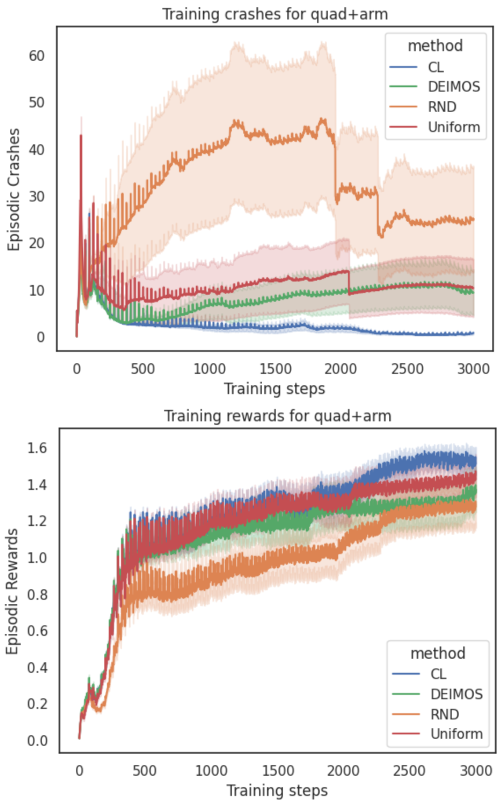

Figure 1: Notice that each training strategy produces very different crash and reward curves: ρ_g has a large influence on training behaviour. (Bear in mind that because they depend on their idiosyncratic ρ_g, **these curves cannot be cross-compared**). Shaded areas are STD errors.

4 Experiments

DEIMOS's fear network is trained using a binary cross-entropy loss on interactions $[< s_t, g_t, g_{t+1} >: W^d_\pi(s_t, g_t, g_{t+1})]$ sampled from π's on-policy rollout buffer. As output, $\mathcal{F}_\Psi$ produces a value from 0 to 1. It is important to note that our fear network is not calibrated. Its only predictive power is a ranking of different goals. We conduct a series of ablations on fear sampling strategies and select the best one (see Appendix 7.1). The policy π_θ is learned using Proximal Policy Optimization [10]. We train and evaluate 30 RL seeds in Legged Gym [4], a state-of-the-art simulator for legged robots based on Isaac Gym [11]. Here, the state s_t contains the robotic state and sensor information and a height map of the surrounding terrain. We study a very unstable morphology [12]: a quadruped with an arm on board ("quad+arm"). The shifting weight of the arm makes traversing steep stairs at fast speeds while also matching desired positional arm goals incredibly difficult. The reward function is primarily based on matching goals but also includes terms to make learning gaits easier [4]; we only report goal-matching rewards. We train using $[\mathcal{T} = 10 \text{ seconds}]$, but to showcase agile behaviour and generalization to faster goal changes, we evaluate using $[\mathcal{T}_2 = 2 \text{ seconds}]$. This timing requires rapidly and responsively handling changes in momentum and bearing the quickly shifting center of weight of the arm. During all evaluations, ρ_g is uniformly sampled from $\mathcal{G}$ to cover all goals in all states indiscriminately. The simulator includes uniformly-noised terrain surfaces and steep stairs and slopes; we evaluate in terrain of "medium" steepness. As the baseline, we use the "Uniform" sampling strategy $\rho_g = \mathcal{U}(\mathcal{G})$. We also evaluate "CL", where the possible range of $\mathcal{G}$ increases as π_θ gets better and where $\rho_g = \mathcal{U}(\mathcal{G})$, as done in Legged Gym [4]. We also implement Random Network Distillation ("RND") over interaction space [13], selecting the most surprising sampled g_{t+1} goal. More results and details can be found in Appendices 7.2 and 7.3.

4.1 Results

As shown in Figure 1, training-goal distributions have a large influence in crashes experienced during training. Figure 2 shows training-goal distributions indeed also have a large influence on deployment crashes, confirming our intuition. Also shown is that DEIMOS trains crash-resistant policies more reliably than the baselines without impacting goal-matching performance (rewards). Additionally, DEIMOS greatly reduces population spread and standard error for crashes in its policy population in the two hardest evaluation environments (while also lowering the mean amount of crashes). Finally, while DEIMOS does not beat the best baseline in the easiest environment and for the quadruped without an arm on board, it remains competitive (for sake of space, tables are shown Appendix 7.3).

5 Related work

Related to the task of improving RL deployment safety are “Safe RL” methods, where explicit safety constraints are applied during training and deployment through reward shaping [1][14], safe intervention/exploration [15][16], or optimization constraints [17][2]. Of particular interest is “Intrinsic Fear” [14], where a fear network is used to shape rewards away from crash-states. Instead, DEIMOS takes inspiration from the constrained optimization field, where “fail-first” methods seek to learn about failure cases before solving a task [18] (this also follows DaGGeR’s intuition [19]: showing recoverable states close to crash-states should enable a policy to learn about recovery). Also closely related to DEIMOS are sub-goal selection methods, some of which train a generative adversarial network (GAN) to generate appropriate goals or environment parameters [7][20][21][22][23][24]. In contrast, we forgo the need to train three models (generator, discriminator, agent) by deriving the generator from the fear network. This avoids the very unstable and hyperparameter-dependant nature of training GANs [21], at the cost of less expressive goal generation (i.e., DEIMOS cannot handle images). Another class of sub-goal selection methods includes Skew-Fit [25], where a uniform distribution over goals is learned (which we empirically show to be beat by DEIMOS for our task).

6 Conclusion

We proposed DEIMOS, a novel training-goal selection method that enables more reliable training of agile robots. In a challenging quadruped locomotion and manipulation task, our method greatly improves trained policy population spread while also either improving or not moving the median and mean compared to the best baseline for both crashes and rewards. In the future, we plan to extend this work, including more RL problems and performance comparisons to other state-of-the-art goal selection methods.

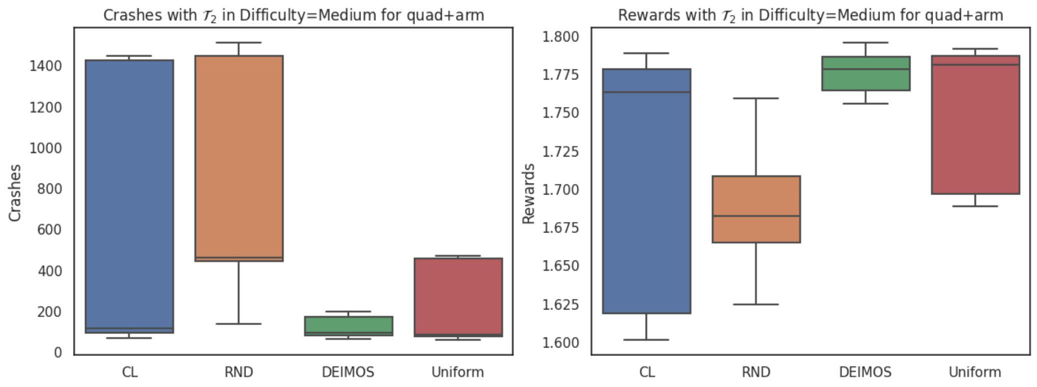

Figure 2: Total evaluation crashes for 30 RL seeds for 8168000 timesteps each: the DEIMOS training regimen is more reliable at training crash-avoiding policies than the baselines. Additionally, average reward performance over the timesteps is near-identical for all methods, with DEIMOS again being the most reliable.

References

[1] A. Sootla, A. I. Cowen-Rivers, T. Jafferjee, Z. Wang, D. H. Mguni, J. Wang, and H. Ammar. Saute RL: Almost surely safe reinforcement learning using state augmentation. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 20423–20443. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/sootla22a.html.

[2] Z. Liu, Z. Cen, V. Isenbaev, W. Liu, Z. S. Wu, B. Li, and D. Zhao. Constrained variational policy optimization for safe reinforcement learning. *CoRR*, abs/2201.11927, 2022. URL https://arxiv.org/abs/2201.11927.

[3] N. L. Kuang and C. H. C. Leung. Performance dynamics and termination errors in reinforcement learning: A unifying perspective. *CoRR*, abs/1902.04179, 2019. URL http://arxiv.org/abs/1902.04179.

[4] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. *CoRR*, abs/2109.11978, 2021. URL https://arxiv.org/abs/2109.11978.

[5] D. Russo, B. V. Roy, A. Kazerouni, and I. Osband. A tutorial on thompson sampling. *CoRR*, abs/1707.02038, 2017. URL http://arxiv.org/abs/1707.02038.

[6] S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. *CoRR*, abs/1111.1797, 2011. URL http://arxiv.org/abs/1111.1797.

[7] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta. Robust adversarial reinforcement learning. *CoRR*, abs/1703.02702, 2017. URL http://arxiv.org/abs/1703.02702.

[8] C. Tessler, Y. Efroni, and S. Mannor. Action robust reinforcement learning and applications in continuous control. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6215–6224. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/tessler19a.html.

[9] P. Zhai, J. Luo, Z. Dong, L. Zhang, S. Wang, and D. Yang. Robust adversarial reinforcement learning with dissipation inequation constraint. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5):5431–5439, Jun. 2022. doi:10.1609/aaai.v36i5.20481. URL https://ojs.aaai.org/index.php/AAAI/article/view/20481.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[11] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning. *CoRR*, abs/2108.10470, 2021. URL https://arxiv.org/abs/2108.10470.

[12] Z. Fu, X. Cheng, and D. Pathak. Deep whole-body control: Learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning (CoRL)*, 2022.

[13] Y. Burda, H. Edwards, A. J. Storkey, and O. Klimov. Exploration by random network distillation. *CoRR*, abs/1810.12894, 2018. URL http://arxiv.org/abs/1810.12894.

[14] Z. C. Lipton, J. Gao, L. Li, J. Chen, and L. Deng. Combating reinforcement learning's sisyphean curse with intrinsic fear. *CoRR*, abs/1611.01211, 2016. URL http://arxiv.org/abs/1611.01211.

[15] S. Junges, N. Jansen, C. Dehnert, U. Topcu, and J. Katoen. Safety-constrained reinforcement learning for mdps. *CoRR*, abs/1510.05880, 2015. URL http://arxiv.org/abs/1510.05880.

[16] N. Wagener, B. Boots, and C. Cheng. Safe reinforcement learning using advantage-based intervention. *CoRR*, abs/2106.09110, 2021. URL https://arxiv.org/abs/2106.09110.

[17] C. Ying, X. Zhou, H. Su, D. Yan, N. Chen, and J. Zhu. Towards safe reinforcement learning via constraining conditional value-at-risk. 2022. doi:10.48550/ARXIV.2206.04436. URL https://arxiv.org/abs/2206.04436.

[18] J. C. Beck, P. Prosser, and R. J. Wallace. Trying again to fail-first. In B. V. Faltings, A. Petcu, F. Fages, and F. Rossi, editors, *Recent Advances in Constraints*, pages 41–55, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-32252-8.

[19] S. Ross, G. J. Gordon, and J. A. Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010. URL http://arxiv.org/abs/1011.0686.

[20] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017. URL http://arxiv.org/abs/1703.06907.

[21] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661.

[22] D. Held, X. Geng, C. Florensa, and P. Abbeel. Automatic goal generation for reinforcement learning agents. *CoRR*, abs/1705.06366, 2017. URL http://arxiv.org/abs/1705.06366.

[23] A. Campero, R. Raileanu, H. Küttler, J. B. Tenenbaum, T. Rocktäschel, and E. Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals. *CoRR*, abs/2006.12122, 2020. URL https://arxiv.org/abs/2006.12122.

[24] L. Weng. Curriculum for reinforcement learning. *lilianweng.github.io*, Jan 2020. URL https://lilianweng.github.io/posts/2020-01-29-curriculum-rl/.

[25] V. H. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine. Skew-fit: State-covering self-supervised reinforcement learning. *CoRR*, abs/1903.03698, 2019. URL http://arxiv.org/abs/1903.03698.

7 Appendix

7.1 Ablations

To select the best goal selection scheme, we conduct a series of ablations upon DEIMOS. (1) Following our Thompson Sampling intuition, "DEIMOS-canonical" selects the most fearful goal in a sample batch. **This version of DEIMOS is purely adversarial.** (2) As verified experimentally (see Subsection 7.3), "DEIMOS-canonical" does a poor job at training π_θ. To address this, we evaluate "DEIMOS-threshold" ("DEIMOS" for short), where random goals above some threshold are selected. In this way, the fear network selects goals that crash the robot "optimistically". (3) To address the issue of the fear network not being calibrated, we evaluate "DEIMOS-min-threshold", where the fear scores are normalized by the minimal fear value of the sample population, and then a random goal over a threshold is selected as in DEIMOS-threshold.

As seen in 7.3, DEIMOS performs better than DEIMOS-canonical and DEIMOS-min-threshold across the board, in every evaluation setup. This version of DEIMOS is the one shown in Section 4.1 and Appendix 7.3.

7.2 Experimental Setup

7.2.1 Training

For the quadruped without an arm on board ("quad"), $\mathcal{G} = [-1, 1]^3$ describes x, y, and angular velocities. For the quadruped with an arm on board ("quad+arm"), $\mathcal{G} = [-1, 1]^6$ describes x, y, and angular velocities and relative x, y and z arm positions.

We use the following reward terms (see Legged Gym [4] for more details):

```
action_rate = -0.01     # penalty on actions
ang_vel_xy = -0.05      # penalty to keep heading straight
base_height = -0.0      # prevents base wobbling when active
collision = -1.0        # prevents collisions
dof_acc = -2.5e-7       # penalty to dof acceleration
dof_vel = -0.0          # penalty to dof velocity
feet_air_time = 1.0     # incentivizes raising feets up
feet_stumble = -0.0     # makes foot movement smoother
lin_vel_z = -2.0        # prevents base wobbling
orientation = -0.0      # penalty to not being upright
stand_still = -0.0      # penalty to no movement
termination = 0.0       # no negative termination term
torques = -0.0002       # penalty to force magnitude
dof_pos_limits = -10.0  # penalty when going over dof limits
tracking_ang_vel = 0.5      # part of goals
tracking_lin_vel = 1.0      # part of goals
tracking_lin_vel_arm = 0.5  # part of goals
```

In addition to the implicit curriculum emerging from the goal selection strategy in play, there is a second curriculum that influences learning in Legged Gym [4]. As the policy learns, the robots it learns over are moved up or down difficulty levels according to the its performance. This is more granular than our "easy", "medium", "hard" evaluation levels; there are 10 levels in all.

7.2.2 Baselines

"Uniform" simply uniformly samples goals from $\mathcal{G}$.

"CL" gradually increases the range of $\mathcal{G}$. This increase happens when the policy matches the requested goals above some threshold. This happens independently for the quadruped-centric goals and for the arm-centric goals.

"RND" is modelled after the intrinsic reward scheme (Random Network Distillation [13]). We apply RND to interaction space. We then invert the RND network in the same manner we invert the fear network, thereby selecting the highest-rated value according to RND. We learn RND using the same interaction experience buffer used to learn DEIMOS' fear network.

7.2.3 Evaluation

We showcase three different evaluation terrain difficulties: "easy", "medium", and "hard", in order of increasing steepness and amplitude applied to noised terrain surfaces. All domain randomization, friction randomization, and random pushes used during training are turned off for evaluation. We also evaluate different goal sampling frequencies ($[\mathcal{T}_1 = 0.5 \text{ seconds}]$, $[\mathcal{T}_2 = 2 \text{ seconds}]$ and $[\mathcal{T}_3 = 10 \text{ seconds}]$). The frequencies were chosen to accurately showcase the policies' response to noise goals ($\mathcal{T}_1$), fast and agile goal changes ($\mathcal{T}_2$), and long-horizon goals ($\mathcal{T}_3$).

All data related to evaluation crashes and goals was subjected to a rolling average with window size 5. The last 10 reported values were then added together. This was done to produce more representative results.

7.3 Supplementary Results

We present the full results of our series of evaluations over many different steepness levels and re-sampling frequencies. For each table, the best performing row is somewhat subjective, as accurately evaluating robotic RL is always difficult. Population standard error is incredibly important. Reducing crashes is obviously desirable, especially within the scope of deploying to real robots, but so is increasing rewards. Often, doing one comes at the cost of the other.

We mark what **we** consider to be the best for each section by "***(1)***" in the method column. Because evaluating robotic RL is so difficult, we also mark the second-best by "***(2)***". We do this because no single method is the best across the board. We find that DEIMOS is often ***(2)*** when it is not ***(1)***. But we are obviously biased, thus we show the full result tables.

7.3.1 Quadruped + Arm

quad+arm in Difficulty=Easy

		Crashes ± STD Error	Rewards ± STD Error
frequency	method		
0.5	CL	621.16 ± 155.44	1.39 ± 0.01
	(2) DEIMOS	109.0 ± 14.28	1.43 ± 0.0
	DEIMOS-canonical	10065.74 ± 2511.32	1.29 ± 0.02
	DEIMOS-min-threshold	118.35 ± 8.43	1.4 ± 0.0
	RND	287.95 ± 60.34	1.41 ± 0.01
	(1) Uniform	40.52 ± 3.68	1.44 ± 0.0
2.0	CL	483.62 ± 120.69	1.77 ± 0.01
	(2) DEIMOS	54.27 ± 7.08	1.81 ± 0.0
	DEIMOS-canonical	9972.31 ± 2488.79	1.65 ± 0.02
	DEIMOS-min-threshold	82.39 ± 5.46	1.78 ± 0.0
	RND	277.9 ± 53.71	1.78 ± 0.01
	(1) Uniform	24.12 ± 2.53	1.82 ± 0.0
10.0	CL	313.27 ± 79.01	1.87 ± 0.01
	(2) DEIMOS	27.32 ± 2.15	1.93 ± 0.0
	DEIMOS-canonical	9861.56 ± 2501.61	1.77 ± 0.02
	DEIMOS-min-threshold	49.75 ± 3.89	1.91 ± 0.0
	RND	9356.39 ± 2152.23	1.8 ± 0.02
	(1) Uniform	20.52 ± 1.26	1.94 ± 0.0

Table 1: DEIMOS beats the other methods, but does worse than Uniform. Thus DEIMOS is best used in harder terrain.

quad+arm in Difficulty=Medium

frequency	method	Crashes ± STD Error	Rewards ± STD Error
0.5	CL	646.98 ± 145.76	1.38 ± 0.01
	(2) DEIMOS	244.07 ± 39.48	1.39 ± 0.01
	DEIMOS-canonical	10900.16 ± 2543.11	1.22 ± 0.02
	DEIMOS-min-threshold	610.77 ± 100.42	1.33 ± 0.01
	RND	552.14 ± 98.03	1.37 ± 0.01
	(1) Uniform	218.69 ± 38.68	1.4 ± 0.01
2.0	CL	546.03 ± 116.9	1.72 ± 0.01
	(1) DEIMOS	120.65 ± 8.75	1.78 ± 0.0
	DEIMOS-canonical	10828.5 ± 2548.06	1.55 ± 0.02
	DEIMOS-min-threshold	731.09 ± 89.98	1.66 ± 0.01
	RND	797.95 ± 103.88	1.69 ± 0.01
	(2) Uniform	209.6 ± 33.72	1.75 ± 0.01
10.0	CL	385.37 ± 75.64	1.83 ± 0.01
	(1) DEIMOS	143.82 ± 9.08	1.87 ± 0.0
	DEIMOS-canonical	10384.7 ± 2526.03	1.67 ± 0.03
	(2) DEIMOS-min-threshold	351.8 ± 42.26	1.8 ± 0.01
	RND	1040.14 ± 198.69	1.74 ± 0.03
	(1) Uniform	153.15 ± 15.7	1.86 ± 0.01

Table 2: DEIMOS does better in all cases except $\mathcal{T} = 0.5$. For $\mathcal{T} = 10$, both DEIMOS and Uniform are marked as *(1)*: they are virtually the same both for crashes and for rewards.

quad+arm in Difficulty=Hard

frequency	method	Crashes ± STD Error	Rewards ± STD Error
0.5	CL	1874.12 ± 158.09	1.21 ± 0.01
	(1) DEIMOS	3274.39 ± 457.38	1.17 ± 0.02
	DEIMOS-canonical	16463.74 ± 2141.75	1.08 ± 0.02
	DEIMOS-min-threshold	7680.41 ± 1082.06	1.01 ± 0.02
	(2) RND	4076.66 ± 737.66	1.22 ± 0.02
	Uniform	4371.62 ± 504.4	1.2 ± 0.01
2.0	CL	2161.77 ± 103.77	1.42 ± 0.01
	(1) DEIMOS	3914.53 ± 484.3	1.4 ± 0.01
	DEIMOS-canonical	17859.94 ± 2487.53	1.28 ± 0.03
	DEIMOS-min-threshold	10069.51 ± 656.95	1.17 ± 0.02
	RND	5557.67 ± 763.41	1.43 ± 0.02
	(2) Uniform	4885.05 ± 773.97	1.42 ± 0.02
10.0	CL	1374.52 ± 98.62	1.58 ± 0.02
	(1) DEIMOS	2527.12 ± 355.95	1.57 ± 0.02
	DEIMOS-canonical	15670.64 ± 3073.54	1.41 ± 0.04
	DEIMOS-min-threshold	5914.04 ± 509.52	1.43 ± 0.02
	RND	4711.6 ± 684.71	1.54 ± 0.01
	(2) Uniform	3636.55 ± 523.42	1.57 ± 0.01

Table 3: DEIMOS also does better here. Note that while CL **seems** to do better, because it performs so much worse in all other environment, we know this to be an artefact of our evaluation strategy. This is why CL is not marked *(1)* in this table.

7.3.2 Quadruped

While impressive in the quadruped+arm setting, DEIMOS is less impressive when applied to a quadruped without an arm on board. This can be explained by the increased instability of the former setting. It is simply harder to issue the right training goals for the bare quadruped. This instability also gives more reason to use DEIMOS; after all, why use a goal sampling more complex than Uniform when the latter is perfectly serviceable?

Finally, notice that possible amounts of crashes for this morphology are much lower. Quad+arm crashes around ten times more often than the bare quadruped, no matter the training regimen. This is why we selected quad+arm for our main evaluation: learning policies for the bare quadruped is already tractable without needing a better goal sampling strategy.

quad in Difficulty=Easy

		Crashes ± STD Error	Rewards ± STD Error
frequency	method		
0.5	CL	58.43 ± 13.7	1.42 ± 0.0
	DEIMOS	14.82 ± 0.57	1.42 ± 0.0
	DEIMOS-canonical	98.81 ± 16.56	1.38 ± 0.0
	DEIMOS-min-threshold	23.04 ± 0.86	1.42 ± 0.0
	(1) RND	6.33 ± 0.75	1.44 ± 0.0
	(2) Uniform	10.74 ± 1.16	1.42 ± 0.0
2.0	CL	29.3 ± 5.48	1.81 ± 0.0
	DEIMOS	15.13 ± 0.86	1.82 ± 0.0
	DEIMOS-canonical	79.53 ± 9.07	1.79 ± 0.0
	DEIMOS-min-threshold	14.56 ± 0.53	1.83 ± 0.0
	(1) RND	6.88 ± 0.5	1.83 ± 0.0
	(1) Uniform	6.57 ± 0.42	1.82 ± 0.0
10.0	***(2)*** CL	4.43 ± 0.09	1.94 ± 0.0
	DEIMOS	8.36 ± 0.6	1.93 ± 0.0
	DEIMOS-canonical	18.45 ± 2.17	1.92 ± 0.0
	DEIMOS-min-threshold	5.88 ± 0.19	1.94 ± 0.0
	RND	5.2 ± 0.24	1.93 ± 0.0
	(1) Uniform	3.94 ± 0.11	1.93 ± 0.0

Table 4: DEIMOS performs competitively but is beat by both Uniform and RND.

quad in Difficulty=Medium

		Crashes ± STD Error	Rewards ± STD Error
frequency	method		
0.5	CL	109.75 ± 16.58	1.39 ± 0.01
	DEIMOS	70.43 ± 2.09	1.39 ± 0.0
	DEIMOS-canonical	249.69 ± 22.1	1.34 ± 0.01
	DEIMOS-min-threshold	77.29 ± 1.63	1.39 ± 0.0
	(1) RND	34.16 ± 1.03	1.41 ± 0.0
	(2) Uniform	43.98 ± 1.43	1.4 ± 0.0
2.0	CL	103.06 ± 4.86	1.77 ± 0.0
	DEIMOS	92.17 ± 2.54	1.77 ± 0.0
	DEIMOS-canonical	226.14 ± 24.79	1.71 ± 0.01
	DEIMOS-min-threshold	94.7 ± 2.09	1.77 ± 0.0
	(1) RND	65.16 ± 0.64	1.79 ± 0.0
	(2) Uniform	72.96 ± 1.7	1.79 ± 0.0
10.0	CL	83.05 ± 2.65	1.89 ± 0.0
	DEIMOS	69.12 ± 0.82	1.89 ± 0.0
	DEIMOS-canonical	147.79 ± 12.28	1.86 ± 0.0
	DEIMOS-min-threshold	80.24 ± 2.51	1.89 ± 0.0
	(2) RND	64.51 ± 0.95	1.89 ± 0.0
	(1) Uniform	62.97 ± 1.0	1.9 ± 0.0

Table 5: Again, RND does fairly well here. DEIMOS is very comparable to both Uniform and RND in most cases.

quad in Difficulty=Hard

		Crashes ± STD Error	Rewards ± STD Error
frequency	method		
0.5	CL	966.91 ± 65.46	1.25 ± 0.01
	DEIMOS	807.47 ± 30.84	1.25 ± 0.01
	DEIMOS-canonical	2658.74 ± 463.67	1.19 ± 0.01
	DEIMOS-min-threshold	535.8 ± 13.49	1.3 ± 0.0
	(1) RND	443.46 ± 22.48	1.31 ± 0.01
	(2) Uniform	490.36 ± 24.95	1.31 ± 0.01
2.0	CL	1146.71 ± 56.07	1.53 ± 0.01
	DEIMOS	909.2 ± 39.49	1.56 ± 0.01
	DEIMOS-canonical	2851.77 ± 550.42	1.46 ± 0.03
	(1) DEIMOS-min-threshold	714.18 ± 4.58	1.64 ± 0.0
	(2) RND	725.64 ± 20.19	1.61 ± 0.01
	(2) Uniform	742.06 ± 12.43	1.59 ± 0.0
10.0	CL	854.23 ± 27.54	1.67 ± 0.01
	DEIMOS	675.52 ± 20.04	1.71 ± 0.01
	DEIMOS-canonical	1858.76 ± 314.89	1.64 ± 0.02
	(2) DEIMOS-min-threshold	614.63 ± 5.98	1.73 ± 0.0
	(1) RND	591.11 ± 5.48	1.75 ± 0.0
	(1)Uniform	602.64 ± 3.43	1.75 ± 0.0

Table 6: Again, Uniform and RND both beat DEIMOS.

7.3.3 Supplementary results discussion

It is very interesting that RND performs so much better in the bare quadruped setting than in the quadruped+arm setting. It seems like the good goal diversity sampled by RND is very useful for training the bare quadruped, but does not lend itself well whatsoever to training the more unstable morphology.

It is also interesting that DEIMOS performs worse here, while DEIMOS-min-threshold performs much better than in quad+arm. This can be explained by the difficulty of learning the fear function. In the bare quadruped setting, the fear network is much easier to learn, and it is thus "more calibrated". In other words, substracting the minimal fear value from the scores influences the scores less, and thus DEIMOS-min-threshold almost catches up to DEIMOS.

Finally, in almost every table presented here, the "rewards" column is the least interesting. All methods achieve remarkably similar reward scores (depending on frequency and difficulty), with only a few outliers (especially in the quad+arm morphology). This is why we selected quad+arm for our main evaluations: quad is simply relatively easy to learn for (at least in Legged Gym [4]).