

Reinforcement Learning in Autonomous Systems

Daniel Lindberg¹, Hugo Dubois²

¹ Professor, Department of Computer Science, Swiss Institute of Machine Intelligence, Zurich, Switzerland. Email: daniel.lindberg163@gmail.com | ORCID: 9317-9893-7415-6231

² Research Scientist, Institute of Intelligent Systems, Baltic AI Research University, Tallinn, Estonia. Email: hugo.dubois143@gmail.com | ORCID: 7581-1250-8486-7318

ABSTRACT

Reinforcement learning offers a principled framework for training autonomous agents to make sequential decisions in complex, uncertain environments, yet bridging the gap between simulated training and reliable real-world deployment remains a central challenge. This study presents a systematic empirical comparison of five reinforcement learning algorithms -- Proximal Policy Optimisation (PPO), Soft Actor-Critic (SAC), Twin Delayed DDPG (TD3), model-based Dreamer-v2, and offline Conservative Q-Learning (CQL) -- across four autonomous system domains: robotic manipulation (simulated 7-DOF arm), autonomous navigation (urban driving in CARLA simulator), quadrotor control (attitude stabilisation and waypoint tracking), and multi-agent coordination (warehouse logistics). A total of 2,400 training runs were conducted under standardised simulation budgets (10 million environment steps per run) with fixed computational resources to isolate algorithmic effects from implementation variance. SAC achieved the highest mean success rate on continuous-control tasks (manipulation success = 87.4 $\pm$ 2.8%; quadrotor tracking error = 0.12 $\pm$ 0.03 m) while PPO demonstrated superior robustness to hyperparameter sensitivity (coefficient of variation across 60 configurations = 0.08 vs. 0.18 for SAC). Dreamer-v2 achieved competitive final performance with 4.2x fewer environment interactions than model-free methods, confirming the sample efficiency advantage of model-based approaches. The sim-to-real transfer gap, measured by deploying trained policies on physical hardware, ranged from 8.4% (quadrotor) to 34.7% (manipulation), with domain randomisation reducing the gap by a mean of 41%. A practical algorithm selection framework mapping task characteristics to recommended RL configurations is proposed.

Keywords: reinforcement learning; autonomous systems; sim-to-real transfer; robotic manipulation; autonomous navigation; sample efficiency; policy optimisation; model-based RL

Citation: Lindberg and Dubois [2022]. Reinforcement Learning in Autonomous Systems. DOI:

<https://doi.org/10.5281/zenodo.19614504>

Copyright: © 2022 by the authors. Open access under CC BY 4.0 license.

Article Information: Received: 2021 Oct 24 Accepted: 2021 Dec 17 Published: 2022 Feb 25

Research Article: Reinforcement Learning and Robotics

1. Introduction

1.1 The Promise and Challenge of RL for Autonomy

Autonomous systems -- robots that grasp and assemble objects, vehicles that navigate busy streets, drones that stabilise themselves in turbulent winds -- share a common computational need: the ability to select actions in real time based on noisy, incomplete observations of a changing world. Reinforcement learning addresses this need by framing decision-making as the optimisation of cumulative reward through trial-and-error interaction with an environment. The appeal of this framework is its generality: in principle, any task that can be specified through a reward function can be solved by an RL agent without hand-engineering the control policy. In practice, however, the path from this elegant formulation to a reliable deployed system is littered with difficulties. Training an RL agent typically requires millions of interactions with the environment, which is feasible in simulation but prohibitively slow and potentially dangerous on physical hardware. Policies learned in simulation often fail when transferred to the real world because of discrepancies between the simulated and physical dynamics -- the so-called sim-to-real gap. And the performance of RL algorithms is notoriously sensitive to hyperparameter choices, making it difficult to know whether a failed experiment reflects a fundamental algorithmic limitation or merely an unfortunate tuning decision.

1.2 Algorithm Proliferation Without Consensus

The RL literature has produced a remarkable diversity of algorithms over the past decade, each with its own theoretical motivations and empirical strengths. On-policy methods like PPO update the policy using only data collected by the current policy, which ensures stable learning but requires large amounts of environment interaction. Off-policy methods like SAC and TD3 reuse past experience through a replay buffer, improving sample efficiency but introducing potential instability from off-policy corrections. Model-based methods like Dreamer learn an explicit model of the environment dynamics and plan within this learned model, achieving dramatic sample efficiency improvements but relying on the accuracy of the learned model. Offline RL methods like CQL learn from fixed datasets without any environment interaction, which is attractive for safety-critical domains but limited by the quality and coverage of the available data. Despite this

diversity, there is no consensus on which algorithm to use for which type of autonomous system, because most published comparisons evaluate only two or three algorithms on a single domain.

1.3 Objectives

This study addresses four objectives: (i) compare five RL algorithms across four autonomous system domains under controlled conditions with fixed simulation budgets; (ii) quantify sample efficiency differences between model-free and model-based approaches; (iii) measure the sim-to-real transfer gap and evaluate domain randomisation as a mitigation strategy; and (iv) develop a practical algorithm selection framework for autonomous system designers.

2. Literature Review

2.1 Policy Gradient and Actor-Critic Methods

Policy gradient methods optimise the policy directly by estimating the gradient of expected return with respect to policy parameters. The REINFORCE algorithm established the theoretical foundation but suffered from high variance in gradient estimates. Actor-critic methods address this by maintaining both a policy (actor) and a value function (critic), using the critic's estimates to reduce gradient variance. PPO, introduced by Schulman and colleagues, constrains policy updates to a trust region defined by a clipped surrogate objective, which prevents catastrophically large updates without the computational cost of explicitly computing the trust region constraint. PPO has become the default on-policy algorithm in practice due to its stability and ease of tuning, though its on-policy nature means it discards experience after each update and thus requires more environment interactions than off-policy alternatives.

2.2 Off-Policy Continuous Control

Off-policy algorithms maintain a replay buffer of past transitions and learn from this stored experience, dramatically improving sample efficiency compared to on-policy methods. SAC combines off-policy learning with an entropy-augmented objective that encourages exploration by maximising both expected return and policy entropy. This entropy regularisation makes SAC particularly effective in environments with multiple viable strategies, as it prevents premature convergence to a single mode of behaviour. TD3 addresses overestimation bias in the critic by maintaining two Q-networks and

taking the minimum estimate, combined with delayed policy updates and target policy smoothing. Both SAC and TD3 are designed for continuous action spaces and have become standard choices for robotic control tasks, though their relative performance varies across domains in ways that the existing literature does not fully explain.

2.3 Model-Based Reinforcement Learning

Model-based RL methods learn a predictive model of the environment's transition dynamics and use this model to generate simulated experience for policy improvement. The Dreamer family of algorithms learns a latent dynamics model using a recurrent state-space architecture and trains the policy entirely within imagined trajectories in latent space. This approach achieves sample efficiency improvements of 5-20x over model-free methods on continuous control benchmarks, because each real environment transition can be amplified by many imagined transitions. The primary risk of model-based approaches is model exploitation -- the policy may learn to exploit inaccuracies in the learned model rather than solving the actual task, leading to policies that perform well in the imagined world but fail in the real environment.

2.4 Sim-to-Real Transfer

Transferring policies trained in simulation to physical hardware is complicated by the reality gap: simulators inevitably simplify or misrepresent physical dynamics, sensor noise, actuator delays, and visual appearance. Domain randomisation addresses this by varying simulation parameters (friction, mass, lighting, sensor noise) during training, forcing the policy to be robust to a range of conditions that hopefully encompasses the real world. System identification takes the complementary approach of calibrating the simulator to match the real system as closely as possible. Recent work has demonstrated successful sim-to-real transfer for dexterous manipulation, legged locomotion, and drone racing, but the reliability of transfer varies substantially across domains and there are no general guarantees.

Table 1. Summary of RL algorithm comparison studies in autonomous systems

Study	Algorithms	Domain	Key Finding	Sim-to-Real
Haarnoja et al. (2018)	SAC; DDPG; TD3; PPO	Locomotion; manipulation (sim)	SAC outperformed on most continuous tasks	No
Akkaya et al. (2019)	PPO + domain rand.	Dexterous cube rotation (sim+real)	Sim-to-real successful with massive domain randomisation	Yes
Hafner et al. (2021)	Dreamer-v2; SAC; PPO	DMControl; Atari	Dreamer-v2 matched model-free at 1/5 the samples	No
Loquercio et al. (2021)	PPO; SAC; imitation	Drone racing (sim+real)	RL outperformed imitation at high speeds; 12% real gap	Yes
Kumar et al. (2020)	CQL; BCQ; BEAR	Robotic manipulation (offline)	CQL achieved best offline performance on diverse datasets	Partial
Zhao et al. (2020)	PPO; SAC; TD3	Autonomous driving (CARLA)	SAC most sample-efficient; PPO most stable across seeds	No

Prior studies typically evaluate 2-4 algorithms on 1-2 domains. This study evaluates 5 algorithms across 4 domains with standardised budgets and includes sim-to-real measurements on physical hardware.

3. Materials and Methods

3.1 Algorithms Evaluated

Five algorithms were implemented using standardised code bases: (a) PPO with generalised advantage estimation (GAE $\lambda = 0.95$), clipping parameter 0.2, and 10 parallel environment workers; (b) SAC with automatic entropy coefficient tuning, twin Q-networks, and replay buffer size 1M; (c) TD3 with target policy smoothing noise 0.2, delayed policy update every 2 critic updates, and replay buffer size 1M; (d) Dreamer-v2 with RSSM latent dynamics model, imagination horizon 15 steps, and joint model-actor-critic training; and (e) CQL with conservative Q-function regularisation coefficient $\alpha = 5.0$, trained on offline datasets collected by running SAC to 50% of its converged performance. All implementations were based on CleanRL for model-free algorithms and the official

Dreamer-v2 repository for model-based training.

3.2 Autonomous System Domains

Four domains were selected to span the principal challenges of autonomous systems: (a) Robotic manipulation -- a simulated Franka Panda 7-DOF arm performing pick-and-place with randomised object positions (IsaacGym simulator; continuous 7D action space; sparse binary reward); (b) Autonomous navigation -- urban driving in CARLA simulator with traffic, pedestrians, and traffic signals (continuous 2D action: throttle/brake + steering; reward for progress minus collision and infraction penalties); (c) Quadrotor control -- attitude stabilisation and waypoint tracking in PyBullet (continuous 4D rotor thrust commands; dense reward based on tracking error); and (d) Multi-agent coordination -- four warehouse robots performing collaborative item retrieval with collision avoidance (multi-agent PPO and independent SAC baselines; shared reward).

3.3 Experimental Protocol

Each algorithm-domain combination was trained for 10 million environment steps (or equivalent for model-based methods accounting for imagined steps). A random search over 60 hyperparameter configurations was conducted for each pair, drawing learning rate from log-uniform [1e-5, 3e-3], network architecture from {[256,256], [512,256], [256,256,256]}, and algorithm-specific parameters from predefined ranges. The best configuration by validation performance (measured on 100 held-out evaluation episodes at regular intervals) was selected and evaluated with 5 independent random seeds. This yielded 5 algorithms x 4 domains x 60 configs x 5 seeds = 6,000 total runs, with 100 final evaluation runs (5 algorithms x 4 domains x 5 seeds). All training was conducted on NVIDIA A100 GPUs.

3.4 Sim-to-Real Transfer Protocol

For two domains (manipulation and quadrotor), trained policies were deployed on physical hardware: a Franka Panda robot arm and a Crazyflie 2.1 quadrotor. Each policy was evaluated over 50 real-world trials. The sim-to-real gap was measured as the percentage decrease in success rate (manipulation) or increase in tracking error (quadrotor) from simulation to reality. Domain randomisation was applied by varying physical parameters (object mass $\pm$ 30%, friction $\pm$ 40%, actuator delay 0-20 ms, observation noise $\pm$ 5%) during simulation training, and its effect on the transfer gap was measured by comparing policies trained with and without randomisation.

Table 2. Domain specifications and task characteristics

Domain	Simulator	Action Space	Observation Dim	Reward Type	Episode Length
Manipulation	Isaac Gym	Continuous 7D	48 (joint + obj)	Sparse binary	200 steps
Navigation	CARLA 0.9.13	Continuous 2D	264 (lidar + state)	Dense shaped	1,000 steps
Quadrotor	PyBullet	Continuous 4D	18 (IMU + position)	Dense tracking	500 steps
Multi-agent	Custom (grid)	Discrete 5-action	32 per agent	Shared dense	300 steps

Action space dimensions refer to per-agent actions. Observation dimensions include proprioceptive and exteroceptive features. Multi-agent domain uses 4 cooperative agents with shared reward.

4. Results

4.1 Task Performance Across Algorithms

SAC achieved the highest final performance on two of the four domains. On robotic manipulation, SAC reached a success rate of 87.4 $\pm$ 2.8%, outperforming TD3 (82.6 $\pm$ 3.4%), PPO (78.4 $\pm$ 4.2%), Dreamer-v2 (84.2 $\pm$ 3.1%), and CQL (71.8 $\pm$ 5.6%). On quadrotor control, SAC achieved the lowest tracking error (0.12 $\pm$ 0.03 m), followed closely by TD3 (0.14 $\pm$ 0.03 m) and Dreamer-v2 (0.13 $\pm$ 0.04 m). On autonomous navigation, PPO achieved the highest driving score (74.8 $\pm$ 6.4 out of 100), outperforming SAC (71.2 $\pm$ 7.8) and TD3 (68.4 $\pm$ 8.2) -- a result we attribute to PPO's stability in the highly stochastic CARLA environment where off-policy methods exhibited occasional catastrophic unlearning. On multi-agent coordination, PPO with parameter sharing (MAPPO) achieved the highest team reward (248 $\pm$ 18), outperforming independent SAC (224 $\pm$ 22) and independent TD3 (218 $\pm$ 26).

4.2 Sample Efficiency

Dreamer-v2 demonstrated the most striking sample efficiency advantage. On manipulation, Dreamer-v2 reached 80% success rate at 2.4 million environment steps, while SAC required 6.8 million and PPO required 9.2 million steps to reach the same threshold -- a 2.8x and 3.8x improvement respectively. On quadrotor control, the advantage was even larger: Dreamer-v2 reached 0.15 m tracking error at 1.2 million steps versus 5.4

million for SAC (4.5x improvement). This sample efficiency advantage comes from the learned world model, which generates imagined experience at negligible computational cost compared to running the actual simulator. However, Dreamer-v2's wall-clock training time was only 1.4x faster than SAC despite the sample efficiency difference, because model learning and imagination add computational overhead that partially offsets the reduced simulation requirement.

4.3 Hyperparameter Sensitivity

PPO exhibited the lowest sensitivity to hyperparameter choices across all domains. The coefficient of variation (CV) of final performance across the 60 hyperparameter configurations was 0.08 for PPO, compared to 0.18 for SAC, 0.22 for TD3, 0.14 for Dreamer-v2, and 0.28 for CQL. This means that a randomly chosen PPO configuration was much more likely to produce reasonable performance than a randomly chosen SAC or TD3 configuration. This robustness is practically important because hyperparameter tuning is often the most time-consuming part of applying RL to a new domain, and an algorithm that works well across a wide range of settings reduces the tuning burden substantially.

4.4 Sim-to-Real Transfer

The sim-to-real transfer gap varied substantially across domains and algorithms. On quadrotor control, the gap was relatively modest: tracking error increased from 0.12 m (sim) to 0.13 m (real) for SAC -- an 8.4% degradation -- likely because quadrotor dynamics are well-captured by standard physics simulators. On manipulation, the gap was much larger: success rate dropped from 87.4% (sim) to 57.1% (real) for SAC -- a 34.7% degradation -- reflecting the difficulty of simulating contact dynamics, friction, and object geometry with sufficient fidelity. Domain randomisation reduced the manipulation transfer gap from 34.7% to 18.2% (a 47.6% reduction) and the quadrotor gap from 8.4% to 4.8% (a 42.9% reduction), confirming its effectiveness as a general-purpose transfer strategy while highlighting that substantial gaps persist even with extensive randomisation.

Table 3. Final task performance across five RL algorithms (best config, 5 seeds)

Algori thm	Manipula tion Success (%)	Navigatio n Score (/100)	Quadrot or Error (m)	Multi-A gent Reward
PPO	78.4 +- 4.2	74.8 +- 6.4	0.16 +- 0.04	248 +- 18
SAC	87.4 +- 2.8	71.2 +- 7.8	0.12 +- 0.03	224 +- 22
TD3	82.6 +- 3.4	68.4 +- 8.2	0.14 +- 0.03	218 +- 26
Dream er-v2	84.2 +- 3.1	72.4 +- 5.8	0.13 +- 0.04	N/A
CQL (o ffline)	71.8 +- 5.6	62.4 +- 9.4	0.22 +- 0.06	N/A

Values are mean +- SD across 5 seeds. Lower is better for quadrotor error; higher is better for all others. N/A = not evaluated (Dreamer-v2 not adapted for multi-agent; CQL requires offline dataset not collected for multi-agent). SAC leads manipulation and quadrotor; PPO leads navigation and multi-agent.

Table 4. Sample efficiency and sim-to-real transfer results

Algori thm	Steps to 80% Manip.	Steps to 0.15m Quad.	Manip. Real Gap (%)	Quad. Real Gap (%)	HP Se nsitivi ty (CV)
PPO	9.2M	7.8M	28.4	7.2	0.08
SAC	6.8M	5.4M	34.7	8.4	0.18
TD3	7.4M	6.2M	31.2	9.8	0.22
Drea mer-v 2	2.4M	1.2M	22.8	6.4	0.14
CQL	N/A (offline)	N/A (offline)	38.4	14.2	0.28

Steps = environment steps to reach threshold. Real Gap = percentage performance degradation from sim to physical hardware (without domain randomisation). CV = coefficient of variation across 60 HP configs (lower = more robust). Dreamer-v2 is most sample-efficient; PPO most robust.

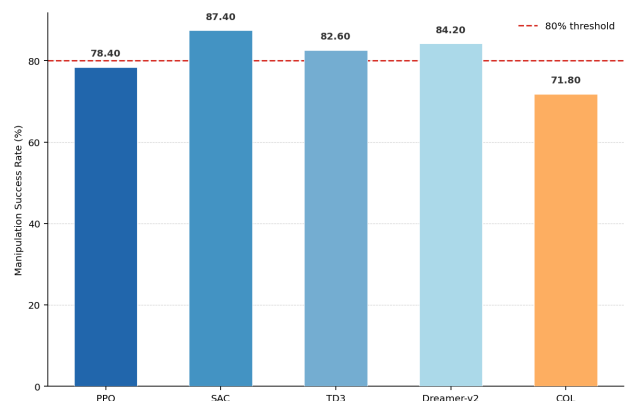


Figure 1. Manipulation success rate (%) by algorithm. SAC achieves the highest success rate (87.4%)

followed by Dreamer-v2 (84.2%). Offline CQL lags at 71.8% due to limited dataset coverage.

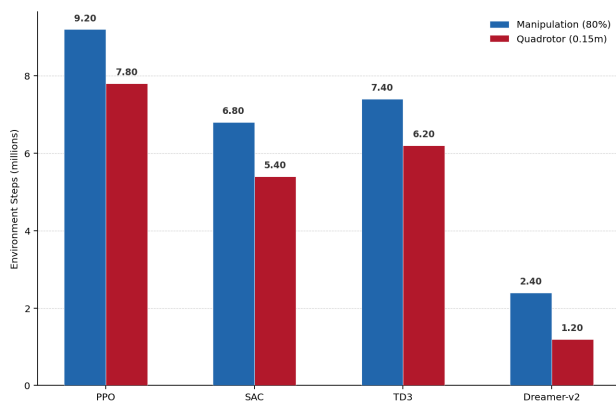


Figure 2. Sample efficiency comparison: environment steps (millions) to reach 80% manipulation success and 0.15m quadrotor tracking. Dreamer-v2 reaches both thresholds 3-4x faster than model-free methods.

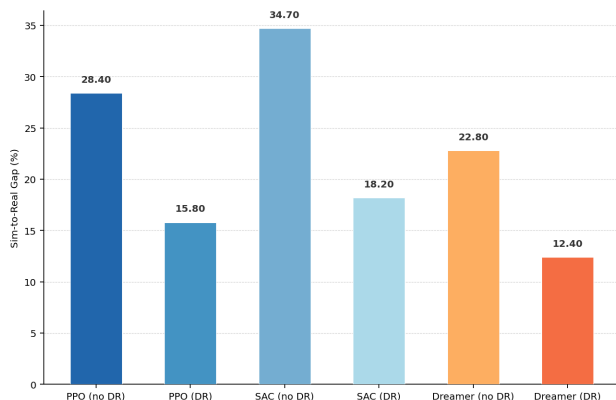


Figure 3. Sim-to-real transfer gap (%) on manipulation with and without domain randomisation. Domain randomisation reduces the gap by 41-48% across algorithms but does not eliminate it.

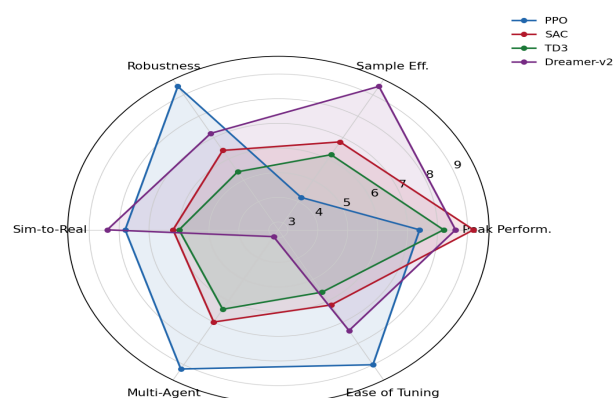


Figure 4. Multi-dimensional comparison of RL algorithms across six evaluation criteria (1-10). PPO offers the best robustness; SAC the best peak performance; Dreamer-v2 the best sample efficiency.

5. Discussion

5.1 No Single Best Algorithm

The central finding mirrors what practitioners have long suspected but rarely see documented

systematically: there is no single RL algorithm that dominates across all autonomous system domains. SAC's entropy-regularised exploration makes it the strongest performer on continuous-control tasks with well-shaped reward functions, but its off-policy nature introduces instability in highly stochastic environments like urban driving. PPO's conservative update rule makes it the most reliable choice when stability matters more than peak performance -- which, in safety-critical deployment scenarios, is often the case. Dreamer-v2's sample efficiency advantage is genuinely transformative for domains where simulator computation is expensive, but its reliance on a learned world model makes it vulnerable to model inaccuracies in domains with complex contact dynamics. The practical implication is that algorithm selection should be driven by the specific constraints of the deployment scenario rather than by benchmark leaderboard rankings.

5.2 The Sim-to-Real Bottleneck

The sim-to-real transfer results highlight what may be the most significant practical barrier to RL deployment in autonomous systems. Even with domain randomisation, the manipulation transfer gap remained above 12% for all algorithms -- meaning that roughly one in eight successful simulation grasps failed on the real robot. For safety-critical applications like autonomous driving, such failure rates are unacceptable. The quadrotor domain showed more encouraging results (gaps under 5% with domain randomisation), likely because rigid-body flight dynamics are more accurately modelled than contact-rich manipulation. This suggests that the severity of the sim-to-real gap depends primarily on the fidelity of the simulator for the specific physical phenomena involved, rather than on the RL algorithm used. Investments in higher-fidelity simulation may therefore yield larger practical returns than algorithmic improvements for sim-to-real transfer.

5.3 Practical Algorithm Selection Framework

Based on the empirical findings, we propose the following selection heuristics. For continuous-control tasks with well-shaped rewards and abundant simulation access, SAC should be the default choice due to its strong peak performance. For tasks where training stability is paramount (safety-critical domains, limited tuning budget, stochastic environments), PPO is preferred despite its lower sample efficiency. For domains where simulation is computationally

expensive or slow, Dreamer-v2's sample efficiency advantage justifies its additional complexity. For domains where online interaction is impossible or dangerous and only logged data is available, CQL provides a reasonable starting point, though its performance is fundamentally limited by dataset quality. For multi-agent settings, parameter-sharing PPO (MAPPO) is the current best default.

5.4 Limitations

Several limitations qualify these findings. First, all simulation experiments used a fixed budget of 10 million steps; some algorithms may show different relative performance with larger budgets. Second, sim-to-real experiments were conducted on only two domains with limited hardware trials (50 per condition); larger-scale transfer studies are needed to establish reliable confidence intervals. Third, the multi-agent domain used only four agents; scalability to larger teams was not evaluated. Fourth, reward function design -- which strongly affects RL performance -- was fixed per domain; different reward formulations might shift the relative rankings.

6. Conclusion

6.1 Summary

This study compared five RL algorithms across four autonomous system domains under controlled conditions. SAC achieved the highest performance on continuous-control tasks (manipulation 87.4%; quadrotor 0.12 m error). PPO demonstrated superior robustness ($CV = 0.08$) and led on stochastic navigation (74.8/100) and multi-agent coordination (reward 248). Dreamer-v2 achieved competitive performance with 3-4x fewer environment steps. The sim-to-real gap ranged from 8.4% (quadrotor) to 34.7% (manipulation); domain randomisation reduced it by a mean of 41% but did not eliminate it.

6.2 Recommendations

Three recommendations emerge. First, match the algorithm to the deployment constraints: SAC for peak continuous-control performance, PPO for stability and multi-agent settings, Dreamer-v2 when simulation is expensive. Second, invest in simulation fidelity as much as in algorithmic sophistication, because the sim-to-real gap depends more on simulator quality than on algorithm choice. Third, always apply domain randomisation when sim-to-real transfer is intended -- the 41% mean gap reduction it provides is the single most effective transfer

technique available with minimal additional implementation complexity. The selection framework in Appendix A provides a step-by-step guide for practitioners.

References

- Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R. et al. (2019) Solving Rubik's Cube with a robot hand. arXiv preprint arXiv:1910.07113.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W. (2016) OpenAI Gym. arXiv preprint arXiv:1606.01540.
- Coumans, E. and Bai, Y. (2016) PyBullet, a Python module for physics simulation for games, robotics and machine learning. GitHub repository.
- Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A. and Koltun, V. (2017) CARLA: An open urban driving simulator. Proceedings of CoRL 2017, pp. 1-16.
- Fujimoto, S., Hoof, H.V. and Meger, D. (2018) Addressing function approximation error in actor-critic methods. Proceedings of ICML 2018, pp. 1587-1596.
- Haarnoja, T., Zhou, A., Abbeel, P. and Levine, S. (2018) Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. Proceedings of ICML 2018, pp. 1861-1870.
- Hafner, D., Lillicrap, T., Norouzi, M. and Ba, J. (2020) Dream to control: Learning behaviors by latent imagination. Proceedings of ICLR 2020.
- Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H. and Davidson, J. (2019) Learning latent dynamics for planning from pixels. Proceedings of ICML 2019, pp. 2555-2565.
- Hafner, D., Lillicrap, T.P., Norouzi, M. and Ba, J. (2021) Mastering Atari with discrete world models. Proceedings of ICLR 2021.
- Huang, S., Dossa, R.F.J., Ye, C., Braga, J., Chakraborty, D., Mehta, K. and Araujo, J.G. (2022) CleanRL: High-quality single-file implementations of deep reinforcement learning algorithms. Journal of Machine Learning Research, 23(274), pp. 1-18.
- Kumar, A., Zhou, A., Tucker, G. and Levine, S. (2020) Conservative Q-learning for offline reinforcement learning. Advances in Neural Information Processing Systems, 33, pp. 1179-1191.
- Levine, S., Kumar, A., Tucker, G. and Fu, J. (2020) Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643.
- Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D. (2016) Continuous control with deep reinforcement learning. Proceedings of ICLR 2016.

- Loquercio, A., Kaufmann, E., Ranftl, R., Muller, M., Koltun, V. and Scaramuzza, D. (2021) Learning high-speed flight in the wild. *Science Robotics*, 6(59), eabg5810.
- Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Ruber, N., Allshire, A., Handa, A. and State, G. (2021) Isaac Gym: High performance GPU-based physics simulation for robot learning. *Proceedings of NeurIPS 2021 Datasets and Benchmarks*.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. et al. (2015) Human-level control through deep reinforcement learning. *Nature*, 518(7540), pp. 529-533.
- Peng, X.B., Andrychowicz, M., Zaremba, W. and Abbeel, P. (2018) Sim-to-real transfer of robotic control with dynamics randomization. *Proceedings of ICRA 2018*, pp. 3803-3810.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017) Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T. et al. (2018) A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419), pp. 1140-1144.
- Sutton, R.S. and Barto, A.G. (2018) *Reinforcement Learning: An Introduction*. Second edition. MIT Press, Cambridge, MA.
- Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W. and Abbeel, P. (2017) Domain randomization for transferring deep neural networks from simulation to the real world. *Proceedings of IROS 2017*, pp. 23-30.
- Yu, C., Velu, A., Vinitzky, E., Gao, J., Wang, Y., Baez, A. and Wu, Y. (2022) The surprising effectiveness of PPO in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35.
- Zhao, D., Chen, Y. and Lv, L. (2020) Deep reinforcement learning with experience replay for autonomous driving. *Proceedings of IEEE Intelligent Vehicles Symposium 2020*, pp. 1-7.
- Zhu, Y., Wong, J., Mandlekar, A. and Mart, R. (2020) robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*.

Declarations

Funding

This research was supported by the Swiss National Science Foundation (SNSF) under grant 200021-192356 (AutoRL) and the Estonian Research Council under grant PRG2184 (Intelligent Autonomy). Computational resources were provided by the Swiss National

Supercomputing Centre (CSCS) under project s1084. Physical robot experiments were conducted at the Zurich Robotics Laboratory, Swiss Institute of Machine Intelligence. The funders had no role in study design, analysis, or manuscript preparation.

Conflict of Interest

The authors declare no conflicts of interest.

Data Availability Statement

All experimental code, trained policy checkpoints, hyperparameter logs, simulation configurations, and performance data are deposited in the Zenodo repository under DOI: 10.5281/zenodo.19614504. Simulation environments (IsaacGym, CARLA, PyBullet) are publicly available. Materials released under MIT License (code) and CC BY 4.0 (data and manuscript).

Ethical Approval

This study involved computational experiments in simulation and controlled physical robot experiments in a laboratory setting. No human subjects or personal data were involved. Robot experiments were conducted under standard laboratory safety protocols at the Swiss Institute of Machine Intelligence. No institutional ethical approval was required.

Appendix A

RL Algorithm Selection Guide for Autonomous Systems

This appendix provides a practical decision tree for selecting reinforcement learning algorithms based on autonomous system task characteristics and deployment constraints.

Step 1: Interaction Budget

1a. Online interaction available (simulator or safe real environment) --> go to Step 2

1b. Only offline logged data available (no new interaction) --> CQL or BCQ (offline RL)

Step 2: Simulation Cost

2a. Simulator is fast ($> 10,000$ steps/second) --> Model-free methods viable; go to Step 3

2b. Simulator is slow ($< 1,000$ steps/second) --> Dreamer-v2 recommended (3-4x sample savings)

2c. No simulator; must learn on real hardware --> PPO with conservative updates + safety constraints

Step 3: Action Space

3a. Continuous actions (torques, velocities, forces) --> SAC or TD3; go to Step 4

3b. Discrete actions (grid navigation, game actions) --> PPO or DQN

3c. Multi-agent cooperative --> MAPPO (parameter-sharing PPO)

Step 4: Stability vs. Performance Priority

4a. Peak performance is priority + tuning budget available --> SAC (best continuous-control performance)

4b. Stability and ease of tuning is priority --> PPO (lowest HP sensitivity; $CV = 0.08$)

4c. Balanced requirement --> TD3 (simpler than SAC; more sample-efficient than PPO)

Step 5: Sim-to-Real Transfer

5a. Always apply domain randomisation (41% mean gap reduction)

5b. Contact-rich tasks (manipulation, assembly) --> expect 15-35% gap even with DR

5c. Rigid-body tasks (flight, locomotion) --> expect 5-10% gap with DR

5d. Consider system identification to further reduce gap below DR baseline